VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro správné generování konfigurací automobilů se závislými komponentami

A Tool For Properly Generating Car Configurations with Dependent Components

2020 Karel Mašlík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Karel Mašlík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Nástroj pro správné generování konfigurací automobilů se závislými komponentami
A Tool for Properly Generating Car Configurations with Dependent Components

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je provést analýzu, návrh a implementaci nástroje pro správnou tvorbu a validaci konfigurací automobilů se závislými komponentami. Nástroj musí nabízet možnost integrace libovolné automobilky i modelu osobních i nákladních vozů.

1. Student analyzuje existující možnosti automobilových konfigurátorů včetně procesu vizualizace výsledné konfigurace vozu.
2. Student analyzuje možnosti automatizovaného získávání aktualizovaných dat o konfiguracích přímo od automobilek.
3. Student navrhne metodu unifikace vlastností automobilů různých výrobců a značek tak, aby se výsledné konfigurace daly jednotně vizualizovat a porovnávat jejich parametry.
4. Výsledkem práce bude analýza, návrh a implementace systému pro tvorbu automobilových konfigurací, včetně návrhu vhodné struktury pro jejich ukládání, vizualizaci a porovnání.
5. Výsledkem bude teoretická příručka, která pomůže při volbě vhodné technologie pro daný projekt s ohledem na výkon, jednoduchost nasazení a finanční náročnost.
6. Teoretické poznatky budou ověřeny na modelovém příkladu implementace minimálně dvou různých prodejců automobilů.

Seznam doporučené odborné literatury:

[1] GORMLEY, Clinton a Zachary TONG. Elasticsearch: the definitive guide. ISBN 1449358543.
[2] SHKLAR, Leon. a Rich. ROSEN. Web application architecture: principles, protocols and practices. 2nd ed. Hoboken, NJ: Wiley, c2009. ISBN 047051860x.
[3] SHIVAKUMAR, Shailesh Kumar. Architecting high performing, scalable and available enterprise web applications. ISBN 9780128022580.
[4] WEERAWARANA, Sanjiva. Web services platform architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more. Upper Saddle River, NJ: Prentice Hall PTR, c2005. ISBN 0131488740.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radoslav Fasuga, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 14. května 2020 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 14. května 2020 .....

Rád bych poděkoval vedoucímu práce Ing. Radoslavu Fasugovi, Ph.D., za ochotu a poskytnutí odborných rad, které byly oporou při tvorbě této práce. Za rady týkající se vývoje struktury konfiguračních dat bych také rád poděkoval svým kolegům. V neposlední řadě děkuji své rodině za poskytnutou podporu při studiu.

Abstrakt

Práce je zaměřena na provedení detailní analýzy jednotlivých prvků automobilových konfiguračních systémů, návrh flexibilního formátu ukládání dat získaných od automobilek a samotnou implementaci konfiguračního systému.

Do analýzy konfiguračních možností bylo zapojeno celkem 20 automobilek, hlavním zaměřením analýzy bylo srovnání jejich konfigurátorů vozidel s ohledem na dostupnou funkcionalitu a ovládací prvky, grafický návrh a využitá data. Z této analýzy vychází návrh struktury dat, která (s určitými nutnými ústupky) umožňuje sjednocení konfiguračních dat automobilek s mnohdy velmi rozdílnými přístupy ke konfiguraci vozů. Navržená struktura dat byla využita za pomoci vytvořeného nástroje, jehož úkolem je stažení potřebných dat přímo od automobilek a jejich následné převedení do zmíněné struktury. V neposlední řadě byla také vytvořena webová aplikace umožňující konfiguraci automobilů včetně validace závislostí jednotlivých komponent.

Přínosem práce je: provedený popis a srovnání existujících řešení využívaných v automobilových konfigurátorech, návrh obecné struktury vhodné pro ukládání konfiguračních dat a také vytvoření algoritmu, který v implementovaném konfigurátoru řeší problematiku změny konfigurace se závislými komponentami.

Klíčová slova: konfigurátor; konfigurace; automobil; komponenty; závislosti

Abstract

This work is focused on perfoming a detailed analysis of individual elements of car configurators, design of a flexible format used for saving data obtained from car manufacturers and implementation of a configuration system.

A total of 20 car manufacturers were part of the analysis of available configuration options. The main focus of the analysis was a comparison of available functionality and control schemes, graphical design and required data of car configurators. The analysis serves as a basis for a design of a data structure, which (with certain necessary compromises) allows for the unification of configuration data of car manufacturers with often very different approaches to vehicle configuration. The designed data structure was used with the help of a created tool used for downloading necessary data directly from car manufacturers and conversion of that data into the aforementioned data structure. Last but not least, a web application allowing for configuring cars and validation of dependencies of individual components was also created.

The contributions of this work are as follows: analysis and comparison of existing solutions used in car configurators, design of a data structure used for saving configuration data and also the creation of an algorithm which solves the issues of configuration changes where components with dependencies are used.

Keywords: configurator; configuration; car; components; dependencies

Obsah

Seznam použitých zkratek a symbolů **10**

Seznam obrázků **11**

Seznam tabulek **12**

Seznam výpisů zdrojového kódu **13**

1 Úvod **14**
1.1 Masová výroba na zakázku . . . 14
1.2 Konfigurace . . . 15
1.3 Konfigurace založená na znalostech . . . 15
1.4 Omezení . . . 15
1.5 Rozdíly mezi obecnou definicí konfigurace a konfiguračními procesy automobilů . 16

2 Analýza konfiguračních možností automobilů **17**
2.1 Typy konfiguračních možností . . . 17
2.2 Analýza konfigurátorů . . . 19
2.3 Konfigurátory užitkových vozů . . . 31
2.4 Řešení kompatibility komponent . . . 33

3 Implementace konfiguračního systému **38**
3.1 Základní struktura konfiguračního systému . . . 38
3.2 Stahování informací o konfiguracích od automobilek . . . 38
3.3 Perzistentní ukládání konfiguračních dat . . . 49
3.4 API zpřístupňující konfigurační data . . . 51
3.5 Webový server zpřístupňující webové stránky . . . 53
3.6 Webová aplikace umožňující konfiguraci automobilů . . . 53

4 Závěr **68**

Literatura **69**

Přílohy **71**

A Zdrojový kód rekurzivní funkce řešící přidávání komponent do aktuální konfigurace **72**

B Zdrojový kód rekurzivní funkce řešící odebírání komponent z aktuální konfigurace **74**

C Teoretická příručka volby technologií pro budování konfiguračních systémů 76
C.1 Úvod . . . 76
C.2 Databáze . . . 76
C.3 Využití cloudových služeb . . . 77
C.4 Doručování statického obsahu . . . 77
C.5 Webové frameworky na straně serveru . . . 78
C.6 Frameworky pro vývoj webových aplikací . . . 79

D Zdrojové kódy a další přílohy 80
D.1 Zdrojový kód nástroje pro stahování konfiguračních dat . . . 80
D.2 Dokumentace zdrojového kódu nástroje pro stahování konfiguračních dat . . . 80
D.3 Příklady zpracovaných konfiguračních dat . . . 80
D.4 Zdrojový kód agregací použitých v MongoDB . . . 80
D.5 Zdrojový kód API zpřístupňující konfigurační data . . . 80
D.6 Zdrojový kód webové aplikace pro konfiguraci automobilů . . . 80
D.7 Dokumentace zdrojového kódu webové aplikace pro konfiguraci automobilů . . . 80

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
BMW	–	Bayerische MotorenWerke
CDN	–	Content Delivery Network
CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
HTTP	–	HyperText Transfer Protocol
JPEG	–	Joint Photographic Experts Group
JS	–	JavaScript
JSON	–	JavaScript Object Notation
MQL	–	MongoDB Query Language
PNG	–	Portable Network Graphics
REST	–	Representational State Transfer
RGB	–	Red Green Blue
SSPL	–	Server Side Public License
URL	–	Uniform Resource Locator

Seznam obrázků

1 Katalog automobilky BMW . . . 19
2 Ukázka principu vizualizace výsledné podoby vozu pomocí kombinace vrstev . . 22
3 Konfigurátor Mitsubishi . . . 23
4 Typický navigační panel . . . 23
5 Konfigurátor značky Opel s atypickým způsobem navigace . . . 24
6 Zobrazení rozdílů stupňů výbavy v konfigurátoru značky Hyundai . . . 25
7 Způsob zobrazení barevných odstínů pomocí předpřipravených ikon . . . 27
8 Způsob vizualizace barevných odstínů pomocí masky . . . 27
9 Způsob vizualizace barevných odstínů pomocí barevného přechodu . . . 27
10 Výběr zdánlivě nekompatibilních položek . . . 29
11 Graf způsobu implementace konfigurátorů užitkových vozů . . . 31
12 Porovnání vzhledu konfigurátorů osobních a užitkových vozů značky Citroën . . 32
13 Výběr provedení užitkového vozu . . . 33
14 Diagram komponent zachycující základní strukturu konfiguračního systému . . . 39
15 Diagram základního konfiguračního modelu . . . 40
16 Diagram struktury tříd uchovávající data o modelu . . . 41
17 Diagram struktury tříd obsahujících údaje o motorizaci . . . 43
18 Diagram popisující strukturu tříd reprezentující konkrétní vozy . . . 44
19 Diagram tříd obstarávajících zpracování dat automobilek . . . 45
20 Sekvenční diagram popisující zpracování konfiguračních dat automobilky Volkswagen 46
21 Návrh rozložení prvků stránky s výběrem stupňů výbavy a motorizací . . . 55
22 Výsledný vzhled hlavních částí konfigurační aplikace . . . 57
23 Shrnutí konfigurace . . . 58
24 Konfigurační dialog . . . 58
25 Diagram aktivit popisující obecný postup konfiguračního procesu . . . 60
26 Struktura hlavních tříd konfigurační aplikace . . . 61
27 Datové třídy využívané v konfiguračním procesu . . . 62
28 Třídní struktury využívané při provádění změn konkrétní konfigurace . . . 62
29 Sekvenční diagram procesu přidání/odebrání komponenty uživatelem . . . 65
30 Pořadí volání a návratů rekurzivních funkcí při vkládání komponenty . . . 67

Seznam tabulek

1 Počet registrací nových osobních automobilů v ČR za rok 2019 18
2 Počty dostupných externích a interních pohledů na výslednou podobu vozu u jednotlivých konfigurátorů 20
3 Datové formáty a pozadí obrázků využívaných při vizualizaci výsledné podoby vozů 21
4 Způsoby ukládání konfigurací v konfigurátorech jednotlivých automobilek 30

Seznam výpisů zdrojového kódu

1 Data přenášená v požadavku na odebrání komponenty v konfigurátoru Škoda . . 34
2 Data přenášená v odpovědi na požadavek odebrání komponenty v konfigurátoru Škoda 35
3 Část dat komponenty z konfigurátoru BMW se závislostmi na ostatních komponentách 37
4 Nastavení spuštění nástroje pro zpracování dat automobilek 47
5 Agregace dat pro výběr modelu automobilky Volkswagen 52
6 Vstupní bod algoritmu generujícího řešení změn konfigurace 66
7 Rekurzivní funkce řešící přidávání komponent do aktuální konfigurace 72
8 Rekurzivní funkce řešící odebírání komponent z aktuální konfigurace 74

1 Úvod

Nabídka základních možností konfigurace produktů není žádnou novinkou. S časem však stále více roste také nabídka produktů, u kterých se konfigurací myslí něco více, než pouhý výběr barvy a typu provedení produktu. [1]

Komplexnější konfigurátory proto ve formě webových aplikací často využívají k umožnění konfigurace potenciálně extrémně velkého množství odlišných produktů závislé komponenty.

Takové komponenty umožňují výrobcům nastavit omezení pro jednotlivé kombinace položek bez nutnosti určování všech nepovolených výsledných konfigurací.

Konfigurátory se těší velké oblibě také v automobilovém průmyslu, kde tyto systémy přináší zákazníkům pocit jedinečnosti a pohodlí při výběru z velkého množství možných barev, výbav, příslušenství atd.

I přesto, že jsou tedy konfigurátory v automobilovém průmyslu poměrně rozšířené, neexistuje mnoho příkladů konfigurátorů, které by umožňovaly kompletní konfiguraci automobilů různých značek.

Snaha o konstrukci takového konfiguračního systému bude jedním z předmětů této práce.

Po základních definicích a obecném přehledu konfiguračních systémů je nutné provést analýzu existujících konfigurátorů vozů z hlediska jejich funkcionality, ovládání a vzhledu.

Poznatky odvozené z této analýzy budou využity při návrhu struktury, která by byla schopna pojmout konfigurační data modelů jakékoli automobilky. Cílem této struktury je tedy umožnění provedení standardizace konfiguračních dat, což také umožní jednodušší porovnání vozů napříč automobilkami.

Navržená struktura bude posléze otestována vytvořením nástroje, jehož úkolem bude získání dat od vybraných automobilek a jejich převod do této struktury.

Konečným cílem bude návrh a implementace zmíněného konfiguračního systému, který bude umožňovat konfiguraci automobilů se závislými komponentami.

Součástí práce je také vytvoření stručné příručky obsahující doporučení technologií vhodných pro budování konfiguračních systémů.

1.1 Masová výroba na zakázku

„Štíhlá výroba je způsob výroby, která kombinuje výhody zakázkové a masové výroby. Tato kombinace umožňuje výrobcům vyvarovat se nevýhodám obou zmíněných typů výroby – vysoké ceně zakázkové výroby a také nepružnosti výroby masové. Jelikož některé části trhu dosahují hranice nasycení, vyvíjí společnosti nová místa na trhu. Zákazníci jsou stále více informovaní a nároční ve svých individuálních nárocích na produkty či služby. Výsledkem těchto dvou sil je změna paradigmatu ve výrobě od masové produkce k masové produkci přizpůsobené potřebám zákazníka.“ [2][3]

Paradigma masové výroby na zakázku se v dnešní době promítá do mnoha různých odvětví. Konfigurací je užíváno jak k personalizaci produktů, tak i služeb. Toho je důkazem databáze

konfigurátorů [1]. Tato sbírka zahrnuje záznamy o 1360 konfigurátorech z nejrůznějších sektorů - od módy, jídla a sportovních potřeb přes stavebnictví až po elektroniku a automobily.

1.2 Konfigurace

Konfiguraci samotnou stručně definovali autoři Sabin a Weigel ve svém článku Product Configuration Frameworks - A Survey [4].

Konfigurace je speciální případ návrhové činnosti se dvěma klíčovými vlastnostmi:

- Konfigurované artefakty jsou složeny z instancí množiny dobře definovaných typů komponent.
- Komponenty mezi sebou interagují předdefinovaným způsobem.

Vybírání a seřazování kombinací jednotlivých částí, které splňují zadané specifikace tvoří jádro konfigurační úlohy. V takovém procesu nemohou být tvořeny žádné nové typy komponent a rozhraní existujících typů komponent nesmí být upravováno. Výsledkem musí být seznam vybraných komponent a také struktura a topologie produktu. [4]

Z důvodu shody označení konfiguračního procesu a jeho výsledku (konfigurace) bude výsledek procesu konfigurace označován jako výsledná konfigurace, popř. konkrétní či aktuální konfigurace.

1.3 Konfigurace založená na znalostech

Konfigurační procesy vázané na znalosti domény konfigurovaných objektů a vzájemného ovlivňování komponent bývají označovány jako konfigurace založené na znalostech. Typy komponent a samotná omezení pak tvoří *konfigurační model*. Konfigurační modely existují jako zobecnění problematiky konfigurace v dané doméně.

To je užitečné kvůli velkému počtu celkových výsledných konfigurací, které by nebylo praktické ukládat kvůli:

- velikosti potřebného úložného prostoru dat,
- času potřebnému k vyhledání výsledné konfigurace splňující požadavky. [5]

1.4 Omezení

Jednotlivé komponenty konfigurací mohou podléhat omezením. Tato omezení ovlivňují možnosti kombinací určitých komponent.

Omezení mohou naznačovat nekompatibilitu komponent, nebo naopak nutnost kombinace s dalšími komponentami. Mezi komponentami mohou také existovat vztahy typu agregace (komponenta obsahuje další komponenty a naopak komponenta je součástí jiné komponenty) a komponenty se řadí do typů - zde se uplatňuje vztah typu generalizace (adaptivní tempomat patří k typu komponent zvaném výbava). [5]

1.5 Rozdíly mezi obecnou definicí konfigurace a konfiguračními procesy automobilů

V obecné konfiguraci založené na znalostech je součástí konfiguračního procesu konfigurační model a veškeré požadavky zákazníka. Tyto části procesu pak slouží jako vstup pro konfigurátor, který vybere konkrétní konfiguraci splňující všechny požadavky zákazníka a vyhovující všem omezením konfiguračního modelu. [5]

U stávajících konfigurátorů automobilů ale součástí počátečního vstupu konfiguračního systému nejsou všechny požadavky zákazníka. Zákazník totiž mnohdy na začátku konfigurace ještě neví, jaké konkrétní vozidlo nebo výbavu bude chtít a jaké přesné parametry má výsledná konfigurace splňovat.

Konfigurátory vozidel proto provázejí zákazníky jednotlivými volbami a zákazník si v průběhu konfiguračního procesu vybírá jednotlivé komponenty. Konfigurátory tak díky volbám zákazníka přechází mezi konkrétními konfiguracemi, které by měly být konzistentní, ale nemusí být kompletní. Jinými slovy, tyto konkrétní konfigurace splňují omezení jednotlivých komponent, ale nemusí splňovat obecná omezení konfiguračního modelu - zákazník si například zatím nezvolil barvu vozidla.

2 Analýza konfiguračních možností automobilů

Pro správný návrh struktury, funkcionality a uživatelského rozhraní konfiguračního systému je nejdříve nutné provést analýzu již existujících řešení různých výrobců. Vzhledem ke specificitě automobilů jakožto konfigurovaných produktů a velmi dobré dostupnosti mnoha různých konfigurátorů budou součástí analyzovaných konfigurátorů pouze konfigurátory motorových vozidel.

Do analýzy bylo zapojeno 20 nejpopulárnějších automobilek seřazených dle počtu registrací nových osobních automobilů v České republice za rok 2019 [6]. Seznam počtů registrací nových osobních vozů těchto automobilek je zachycen v tabulce 1. Jakékoli pozdější zmínky analyzovaných automobilek, jejich konfigurátorů či jejich počtu se budou vztahovat k automobilkám uvedeným v této tabulce. Analyzovány byly vždy oficiální webové stránky a konfigurátory určené pro český trh. Pokud má automobilka různé konfigurátory pro osobní a užitkové vozy, v analýze je odkazováno na konfigurátory pro osobní vozy, pokud není uvedeno jinak. Porovnání konfigurátorů pro osobní a užitková vozidla je dostupné v sekci 2.3.

2.1 Typy konfiguračních možností

Automobilky využívají pro poskytování informací o dostupnosti kombinací modelů, motorizací, barev a dalších možností výběrů vlastností vozidel různé prostředky.

2.1.1 Katalogy a ceníky

Jedním z takových prostředků, vyskytující se u všech analyzovaných automobilek, jsou katalogy (popř. ceníky) vozidel ve formátu PDF. Tyto katalogy poskytují informace o ceně jednotlivých motorizací, barev, kol, interiérů a samozřejmě také doplňkové výbavy. Dále informují o dostupnosti jednotlivých možností v závislosti na zvolené motorizaci či stupni výbavy.

Některé z automobilek skrze tyto katalogy a ceníky dokonce zveřejňují omezení výběru jednotlivých položek výbavy v závislosti na výběru dalších položek. Příkladem automobilky zveřejňující tyto informace ve svých katalozích je BMW, ukázka katalogu BMW je na obrázku 1.

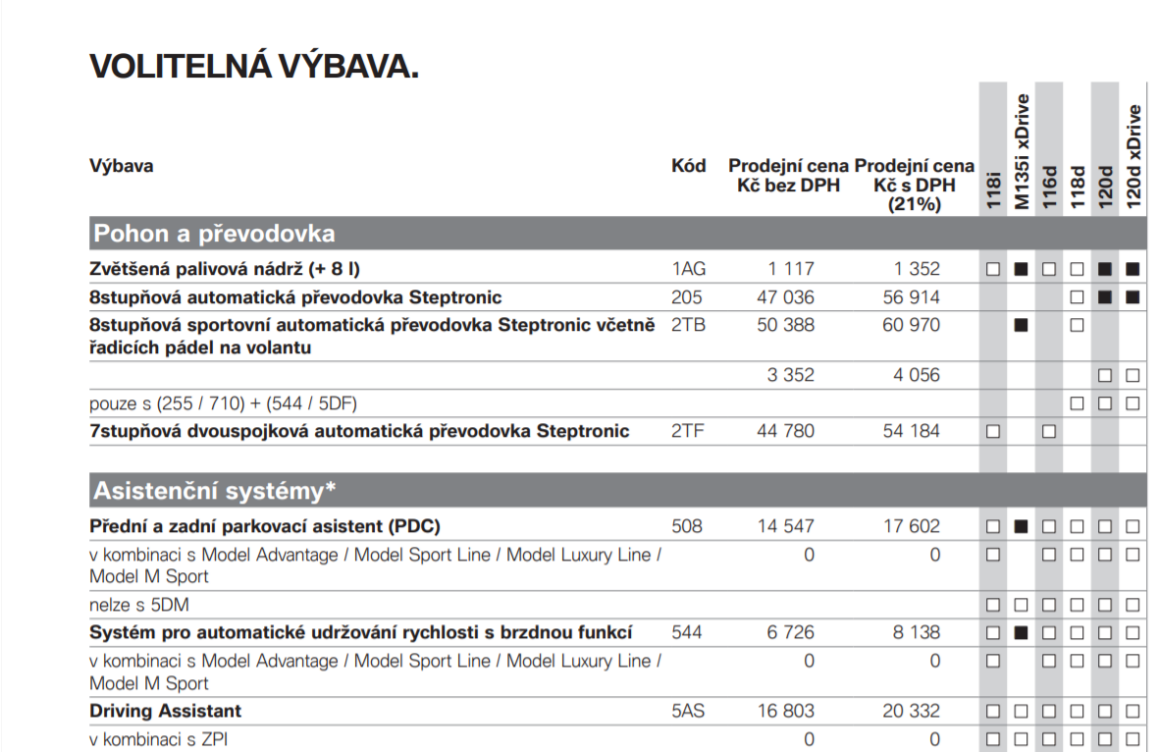
2.1.2 Konfigurátory v podobě webových aplikací

Další možností poskytnutí informací o dostupných možnostech konfigurací vozidel jsou samozřejmě konfigurátory ve formě webových aplikací. Na rozdíl od katalogů poskytují konfigurátory uživateli:

- velkou míru interaktivity,
- okamžitou vizuální zpětnou vazbu při vybírání barev, kol a interiérů vozidel díky vizualizaci přibližné konečné podoby vozidla,
- větší komfort - např. při počítání výsledné ceny nebo při kontrole kompatibility komponent.

Tabulka 1: Počet registrací nových osobních automobilů v ČR za rok 2019 [6]

Pořadí	Název automobilky	Počet vozů	Podíl na celkovém počtu
1.	Škoda	85895	34,37%
2.	Volkswagen	20869	8,35%
3.	Hyundai	19302	7,72%
4.	Dacia	14807	5,92%
5.	Peugeot	11346	4,54%
6.	Renault	10101	4,04%
7.	Toyota	9893	3,96%
8.	Ford	9739	3,90%
9.	Kia	9681	3,87%
10.	Mercedes-Benz	7322	2,93%
11.	Seat	6671	2,67%
12.	Citroën	6552	2,62%
13.	BMW	5386	2,16%
14.	Opel	4713	1,89%
15.	Suzuki	4343	1,74%
16.	Mazda	4086	1,63%
17.	Audi	2600	1,04%
18.	Volvo	2334	0,93%
19.	Mitsubishi	2246	0,90%
20.	Honda	2071	0,83%
Celkový podíl automobilek			96,01%



VOLITELNÁ VÝBAVA.

Výbava	Kód	Prodejní cena Kč bez DPH	Prodejní cena Kč s DPH (21%)	118i	M135i xDrive	116d	118d	120d	120d xDrive
Pohon a převodovka									
Zvětšená palivová nádrž (+ 8 l)	1AG	1 117	1 352	□	■	□	□	■	■
8stupňová automatická převodovka Steptronic	205	47 036	56 914				□	■	■
8stupňová sportovní automatická převodovka Steptronic včetně řadicích pádel na volantu	2TB	50 388	60 970		■		□		
		3 352	4 056					□	□
pouze s (255 / 710) + (544 / 5DF)							□	□	□
7stupňová dvouspojková automatická převodovka Steptronic	2TF	44 780	54 184	□		□			
Asistenční systémy*									
Přední a zadní parkovací asistent (PDC)	508	14 547	17 602	□	■	□	□	□	□
v kombinaci s Model Advantage / Model Sport Line / Model Luxury Line / Model M Sport		0	0	□		□	□	□	□
nelze s 5DM				□	□	□	□	□	□
Systém pro automatické udržování rychlosti s brzdnou funkcí	544	6 726	8 138	□	■	□	□	□	□
v kombinaci s Model Advantage / Model Sport Line / Model Luxury Line / Model M Sport		0	0	□		□	□	□	□
Driving Assistant	5AS	16 803	20 332	□	□	□	□	□	□
v kombinaci s ZPI		0	0	□	□	□	□	□	□

Obrázek 1: Katalog automobilky BMW znázorňující dostupnost výbavy v závislosti na motorizaci a ostatních položkách výbavy (www.bmw.cz)

Z analyzovaných automobilek neposkytuje webovou aplikaci pro konfiguraci automobilů pouze jediná automobilka, a to Suzuki. Oficiální stránka Suzuki pro německý trh však jednoduchý konfigurátor poskytuje.

Webové konfigurátory vozů používají data získaná skrz API automobilek ve formátu JSON. Tento zdroj informací o vozidlech a jejich konfiguračních možnostech umožňuje na rozdíl od ceníků a katalogů jednodušší strojové čtení a manipulaci s přečtenými daty.

2.2 Analýza konfigurátorů

Konfigurátory jednotlivých automobilek se od sebe liší v mnoha ohledech. Popis rozdílů a příklady konkrétních implementací hlavních částí konfigurátorů se nachází níže.

2.2.1 Vizualizace podoby vozidla

Vizualizační prvky konfigurovaných automobilů jsou u automobilek různorodě implementovány, v drtivé většině konfigurátorů však tvoří dominantní prvek stránky.

Nejčastěji je jako forma vizualizace využívána galerie obrázků s různými úhly pohledu na vozidlo. Každá automobilka nabízí různý počet pohledů na konfigurované vozidlo jak zvenku, tak zevnitř. U některých automobilek (např. Peugeot) se počet snímků a typ dostupných pohledů na vozidlo mění v závislosti na konkrétním modelu.

Tabulka 2 ukazuje standardní počty obrázků s pohledy na exteriér a interiér vozidla dostupných uživatelům konfigurátorů jednotlivých automobilek. V tabulce jsou také vyznačeny zvláštnosti vizualizačních prvků analyzovaných konfigurátorů osobních automobilů.

Tabulka 2: Počty dostupných externích a interních pohledů na výslednou podobu vozu u jednotlivých konfigurátorů s případnými vyznačenými zvláštnostmi implementace

Automobilka	Pohledů na exteriér	Pohledů na interiér
Škoda	3	3
Volkswagen	7	2
Hyundai	1	1 miniatura
Dacia	24 (po 15°)	360° (složen ze 6 snímků)
Peugeot	5	3
Renault	24 (po 15°)	360° (složen ze 6 snímků)
Toyota	36 (po 10°)	3
Ford	9	4
Kia	2	1 miniatura
Mercedes-Benz	36 (po 10°, denní a noční provedení)	4
Seat	24 (po 15°)	3
Citroen	5	3
BMW	36 (po 10°, denní a noční provedení)	2
Opel	4	3
Mazda	36 (po 10°)	360° (1 panoramatický snímek)
Audi	7	2
Volvo	5	3
Mitsubishi	6	2 miniatury
Honda	2 (denní a noční provedení)	3

Nezanedbatelný počet automobilek (7 z 19) využívá pro vizualizaci exteriéru automobilu sekvenci snímků zachycených z úhlů pohledů rovnoměrně rozmístěných po kružnici okolo vozidla. Uživateli je tak nabízen pohled na vozidlo ze všech úhlů bez nutnosti stahování a zobrazování 3D modelu automobilu.

Pro zobrazení interiéru využívají automobilky Dacia, Renault a Mazda 360stupňového pohledu. U automobilek Dacia a Renault je tento pohled realizován sešitím 6 samostatných snímků, Mazda tento pohled generuje pomocí jednoho panoramatického snímku.

Automobilky Mercedes-Benz, BMW a Honda nabízejí všechny obrázky s externími pohledy na vozy ve dvou provedeních: standardní obrázky simulují výslednou podobu automobilu

za světla a druhá sada obrázků simuluje podobu vozů s rozsvícenými světlomety v noci.

Datové formáty obrázků využívaných ve vizualizačních prvcích jednotlivých konfigurátorů a způsoby provedení pozadí u těchto obrázků jsou k vidění v tabulce 3.

Tabulka 3: Datové formáty a pozadí obrázků využívaných při vizualizaci výsledné podoby vozů

Značka	Formát (exteriér)	Formát (interiér)	Pozadí (exteriér)	Pozadí (interiér)
Škoda	PNG	JPEG	bez pozadí	bílošedý přechod
Volkswagen	JPEG	JPEG	bílé	šedé
Hyundai	PNG	JPEG	bez pozadí	šedé
Dacia	JPEG	JPEG	bílé	bílé
Peugeot	JPEG	JPEG	bílé	bílé
Renault	JPEG	JPEG	bílé	bílé
Toyota	JPEG	JPEG	šedé	šedé
Ford	WebP	WebP	bílé	šedé
Kia	PNG	JPEG	bez pozadí	bílošedý přechod
Mercedes-Benz	WebP	JPEG	interiér budovy	interiér budovy
Seat	PNG	PNG	bez pozadí	bílošedý přechod
Citroën	JPEG	JPEG	bílé	bílošedý přechod
BMW	JPEG	JPEG	šedé geometrické tvary	bílošedý přechod
Opel	JPEG	JPEG	žlutý pruh	bílošedý přechod
Mazda	JPEG	JPEG	šedočerný přechod	šedé
Audi	PNG	PNG	bez pozadí	bez pozadí
Volvo	JPEG	JPEG	bílé	bílé
Mitsubishi	PNG	JPEG	bez pozadí	bílé
Honda	PNG	JPEG	bez pozadí	město

Lze vidět, že automobilky obecně preferují formát JPEG - 10 z 19 konfigurátorů využívá tento formát pro zobrazení exteriéru vozů a 16 z 19 konfigurátorů jej používá pro zobrazování interiérů.

U automobilek Ford a Mercedes-Benz se také vyskytuje formát WebP, který podporuje ztrátovou i bezztrátovou kompresi. Formát dále také podporuje průhlednost. [7] „Oproti formátu PNG nabízí formát WebP lepší kompresní poměr. V kompresním poměru a kvalitě obrazu také WebP překonává formát JPEG za cenu dlouhého času potřebného k zakódování obrazu.“ [8]

Bezztrátový kompresní formát PNG bývá využíván zejména kvůli dostupnosti alfa kanálu, který umožňuje nejenom použití na jakémkoli pozadí stránky, ale u konfigurátorů značek Kia

a Honda je tento formát rovněž využit k překrývání základního obrázku vozidla dalšími obrázky reprezentujícími jednotlivé volitelné části. Tento princip je znázorněn pomocí obrázků 2a-2d. Na základ (obrázek 2a) jsou přidávány další vrstvy - např. vrstvy barvy a výbavy (na obrázku 2c lze vidět vrstvu reprezentující změnu vzhledu způsobenou přidáním zatmavených skel). Uživateli je pak zobrazen výsledný obrázek vytvořený spojením všech těchto vrstev (obrázek 2d).

(a) základní vrstva

(b) vrstva barvy

(c) vrstva zatmavených oken

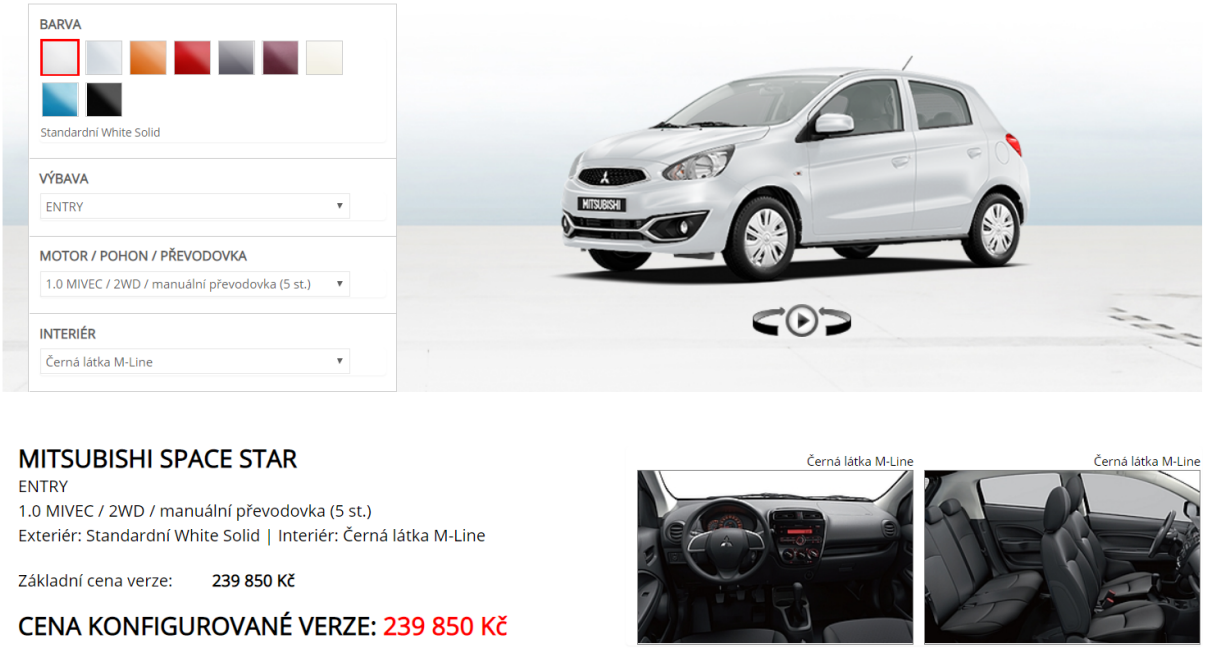
(d) výsledný obrázek

Obrázek 2: Ukázka principu vizualizace výsledné podoby vozu pomocí kombinace vrstev (www.kia.com/cz)

Automobilky většinou volí jednoduchá světlá pozadí snímků s neutrálními barvami (bílá, šedá a bílošedý přechod). Tomu se vymykají pouze automobilky Mercedes-Benz, BMW, Opel, Mazda a Honda. Jednoduchá neutrální pozadí neubírají pozornost od vzhledu vozidla a navíc umožňují jednoduché sladění s jakýmkoli grafickým návrhem konfigurátoru.

2.2.2 Navigace a rozdělení do sekcí

Analyzované konfigurátory nabízejí uživatelům konfiguraci automobilu rozdělenou do několika částí. Tyto části jsou od sebe ve velké většině konfigurátorů kompletně odděleny celkovou výměnou obsahu stránky. Výjimku donedávna představoval konfigurátor Mercedes-Benz, který používal jednostránkový grafický návrh pro konfiguraci svých vozů, kde byly jednotlivé části konfigurace odděleny pouze horizontální hranicí. Dalším příkladem je jednoduchý konfigurátor Mitsubishi (obrázek 3), který obsahuje pouze základní možnosti konfigurace.



Obrázek 3: Konfigurátor Mitsubishi (www.mitsubishi-motors.cz)

Přístup k jednotlivým částem konfigurace je u většiny konfigurátorů možný přes velká navigační tlačítka umístěná typicky na levé či horní straně obrazovky. Tato tlačítka také signalizují zákazníkův postup konfigurací a vzdálenost zákazníka od závěrečného shrnutí. Typický příklad takového navigačního panelu nabízí konfigurátor značky Renault (obrázek 4). Ostatním konfigurátorům se svým způsobem navigace vymyká konfigurátor značky Opel, který je koncipován jako průvodce pouze s kroky další a zpět. Nenabízí tak celkovou mapu konfigurace (obrázek 5).

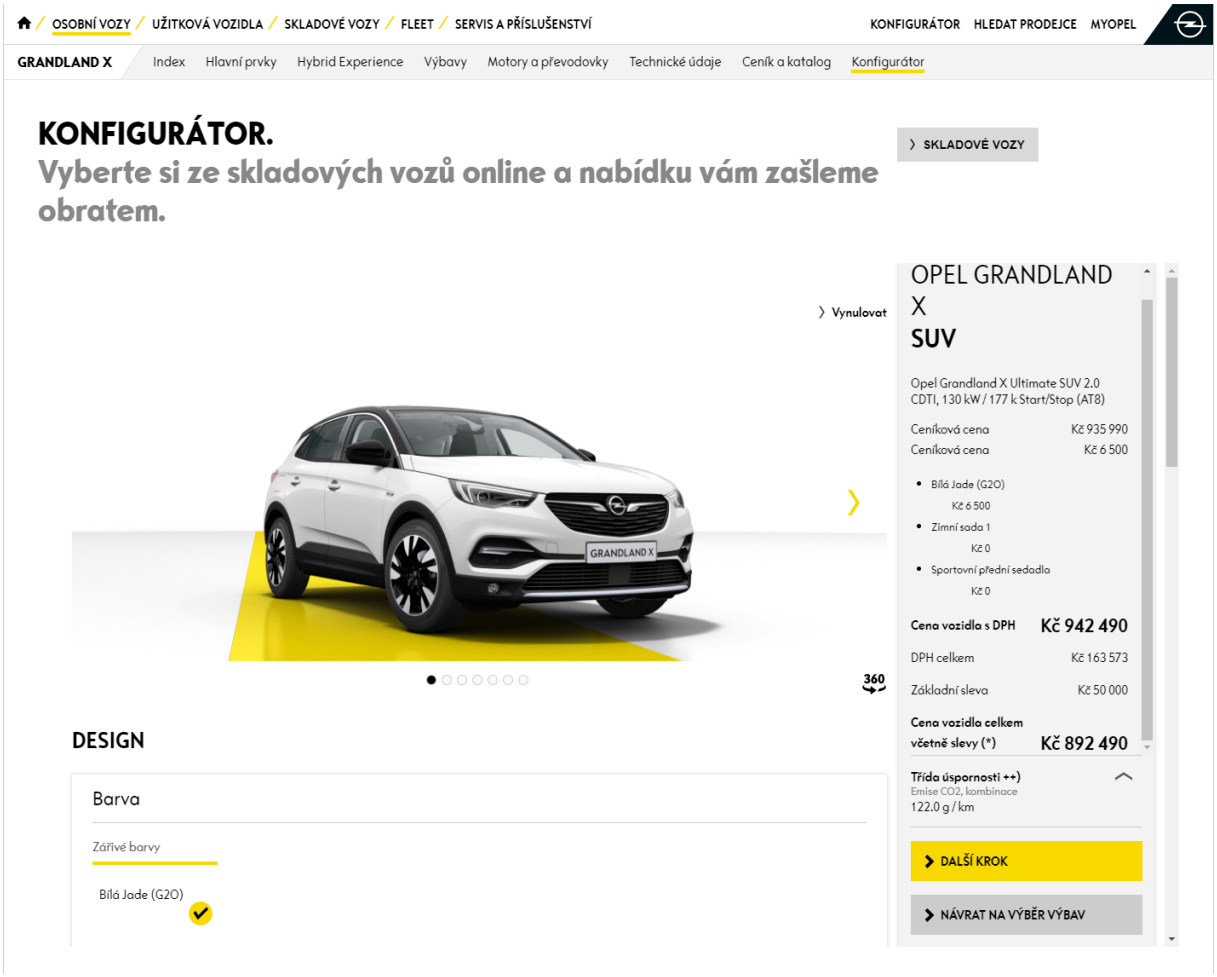
Obrázek 4: Typický navigační panel (www.renault.cz)

Samotné sekce konfigurátorů a pořadí, ve kterém uživatele provází konfigurací, se dají obecně shrnout následovně: výběr modelu (většinou není přímou součástí samotné konfigurační aplikace), výběr stupně výbavy, motorizace, barvy vozidla, kol, interiéru, příplatkové výbavy a závěrečné shrnutí.

Opět samozřejmě existují výjimky:

- konfigurátor Škody má zařazený výběr motorizace až za výběr interiéru,
- konfigurátor značky Hyundai zařazuje výběr výbavy před výběr barvy a interiéru,
- Peugeot, Ford a Citroën mají výběr kol zařazený do výběru příplatkové výbavy,

další výjimky týkající se jednotlivých sekcí jsou uvedeny v popisu těchto sekcí.



Obrázek 5: Konfigurátor značky Opel s atypickým způsobem navigace (www.opel.cz)

2.2.3 Výběr modelu

První krok uživatelů začíná výběrem modelu automobilu. U většiny automobilek si uživatel v tomto kroku vybírá z několika modelových řad, které jsou ještě rozděleny dle typu karoserie (např. Škoda Fabia a Škoda Fabia Combi jsou v konfigurátoru uvedeny jako 2 samostatné položky). Výjimkou je konfigurátor značky Ford, ve kterém je uživatelům na úvodní stránce nabídnut seznam modelových řad a výběr karosérie je uživateli umožněn až jako pozdější krok konfigurace automobilu.

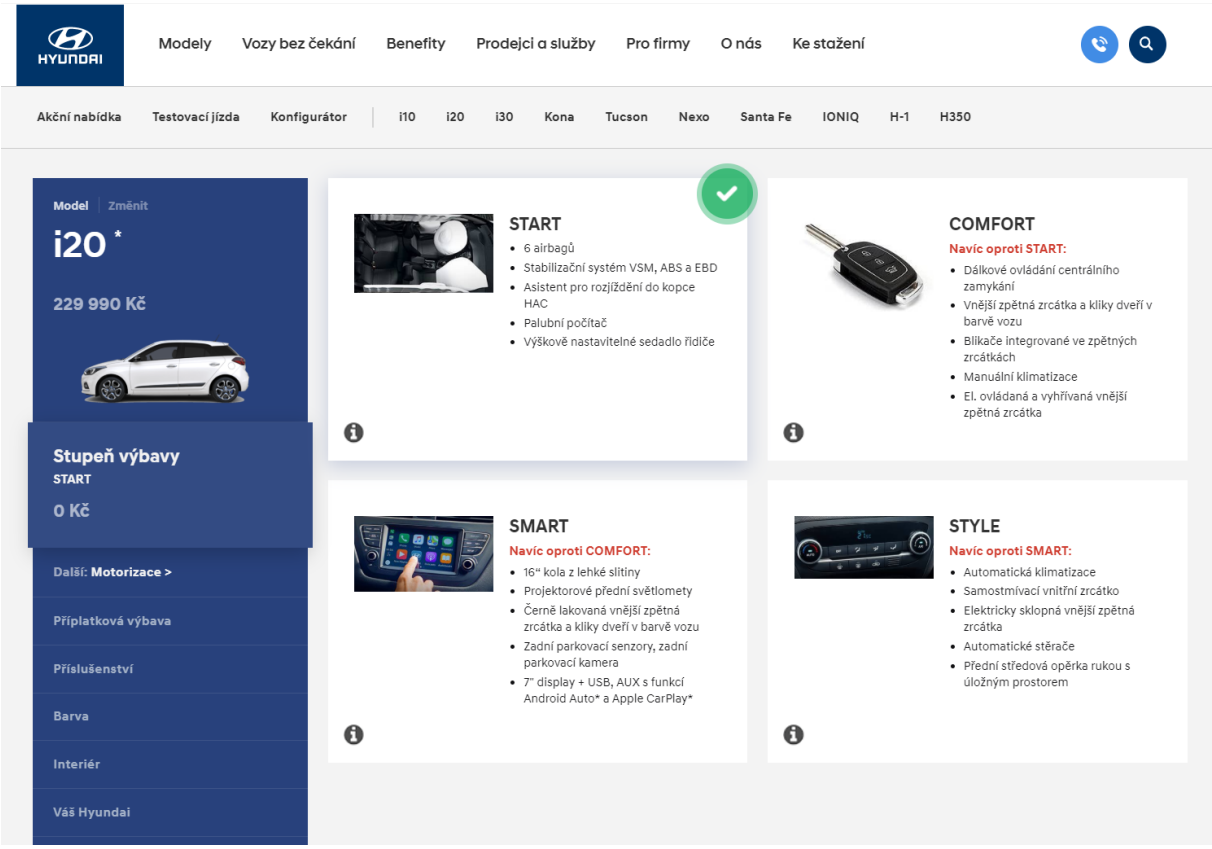
Při výběru modelu zákazníkům nabízejí automobilky filtry pro zúžení výběru dostupných modelů, nejčastěji se jedná o omezení výběru podle typu karosérie nebo podle základní ceny.

2.2.4 Výběr stupně výbavy

Analyzované konfigurátory využívají 2 hlavní principy zobrazení výčtů vybavení obsaženého u konkrétních stupňů výbavy.

1. Veškerá obsažená výbava (popř. její nejdůležitější součásti) je zobrazena nezávisle na ostatních dostupných stupních výbavy.

2. Stupně výbavy jsou zákazníkům představovány jako výčty vybavení, které daný stupeň výbavy obsahuje navíc oproti nižšímu stupni výbavy. Příklad tohoto způsobu zobrazení stupňů výbavy je na obrázku 6. Pro použití tohoto způsobu zobrazení musí být stupně výbavy koncipovány podle následujícího pravidla: nižší stupeň výbavy musí být podmnožinou vyššího stupně výbavy. Toto pravidlo však neplatí u mnoha automobilek - dobrým příkladem výbavy mnohdy porušující toto pravidlo jsou například kola. Vyšší stupeň výbavy často nezahrnuje kola menší velikosti dostupné v nižším stupni výbavy.



Obrázek 6: Zobrazení rozdílů stupňů výbavy v konfigurátoru značky Hyundai (www.hyundai.cz)

Výběr stupně výbavy není stejný u všech konfigurátorů automobilek. Například u některých modelů Mercedes-Benz je nutné provést výběr 2 stupňů výbav - jeden stupeň výbavy pro exteriér a druhý pro interiér. Naopak u značky BMW není tato sekce dostupná u všech modelů. V takovém případě má zákazník na výběr z motorizací dostupných pro daný model a dostupnost výbavy je vázána na vybranou motorizaci.

2.2.5 Výběr motorizace

Výběr motorizace je ve většině případů krok přímo následující výběr stupně výbavy. Výjimky tvoří:

- konfigurátor Škoda, kde je výběr motorizace zařazen až za výběr stupně výbavy, barvy, kol a interiéru,
- konfigurátory Mercedes-Benz a BMW, u kterých výběr motorizace předchází výběr stupně výbavy,
- nebo například konfigurátor značky Honda, kde záleží na výběru uživatele, zda chce začít výběrem motorizace či výběrem stupně výbavy.

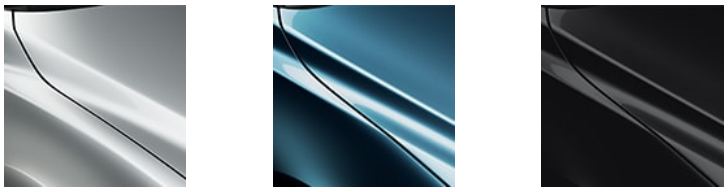
Standardní u většiny konfigurátorů je doplnění výběru motorizace filtry, které pomáhají uživateli omezit výběr motorizace dle typu paliva a převodovky, popř. podle pohonu kol.

Typ a počet technických informací vozů dostupných při výběru motorizací jsou napříč konfigurátory velmi odlišné. Kromě údajů týkajících se typu paliva, typu převodovky, pohonu kol a základní ceny motorizace jsou nejčastěji zobrazovány pouze údaje o spotřebě paliva, hodnoty emisí a výkonu motoru (udávaný v koních nebo kilowattech).

2.2.6 Výběr barvy

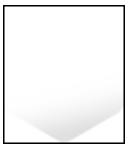
Vizualizace barevných odstínů je v analyzovaných konfigurátorech realizována jedním ze tří způsobů.

1. Nejčastěji využívaným způsobem je zobrazování odstínů pomocí předpřipravených ikon barev. S tímto způsobem zobrazení odstínu je možné zachytit odlesky na specifických lomech materiálů daných modelů vozidel (obrázek 7).
2. Druhým způsobem je zobrazení barvy pomocí RGB kódu a následné překrytí barevného elementu pomocí speciálního obrázku ve formátu PNG, který vytváří dojem odlesku barvy. Tento princip je jak praktický, tak i efektní - jednu masku lze použít u více barev a zároveň každý typ barvy (lesklá, metalická) může využívat svého vlastního efektu odlesku. Tento způsob zobrazování barev využívá ve svém konfigurátoru například automobilka Honda (obrázek 8).
3. Poslední způsob také využívá hodnoty barvy namísto předpřipravených ikon, element je však v tomto případě upraven s pomocí barevného přechodu za využití kaskádových stylů. Tento způsob vizualizace barevných odstínů může být považován za nejméně přesný - odstín je spíše orientační. Toto však u konfigurátorů se zobrazováním konečné podoby automobilu nepředstavuje problém. Výhodou je také obecné využití bez nutnosti přípravy ikon či obrázků určených k překrývání barevných elementů. Tento způsob zobrazování barev je využit v konfigurátoru značky Škoda (obrázek 9).



Obrázek 7: Způsob zobrazení barevných odstínů pomocí předpřipravených ikon (www.mazda.cz)

(a) maska odstínu na černém pozadí



(b) maska odstínu na bílém pozadí

(c) výsledná ikona barvy

Obrázek 8: Způsob vizualizace barevných odstínů pomocí masky (www.honda.cz)

Obrázek 9: Způsob vizualizace barevných odstínů pomocí barevného přechodu (cc.skoda-auto.com/cze/cs-CZ)

Barvy exteriéru vozů jsou u většiny analyzovaných konfigurátorů rozděleny do skupin. Nejčastěji je toto rozdělení provedeno na základě typů barev. Napříč automobilkami se objevují následující typy barev: základní, metalické, perleťové, speciální (např. historicky významné odstíny automobilky) a kombinace barev (např. dvojice barev, kde je jedna barva určena pro střechu a druhá pro zbytek vozidla). U automobilek Dacia a Renault se také vyskytuje dělení barev do skupin dle cenové kategorie.

2.2.7 Výběr kol

Kola jsou v konfigurátorech nejčastěji seskupena dle velikostí, méně časté je pak seskupení podle materiálů, ze kterých jsou kola vyrobena.

Některé konfigurátory výběr kola vizuálně podpoří i jinými prvky než standardními miniaturami. Například konfigurátory Renault a Dacia při výběru kol přepínají aktivní obrázek náhledu automobilu na detailní pohled na oblast předního kola.

2.2.8 Výběr interiéru

Na rozdíl od sekcí výběru barev a kol, nejsou sekce s interiérovými položkami většinou rozdělovány do skupin. U zbývajících konfigurátorů se většinou jedná o seskupení na základě použitých materiálů.

Některé automobilky v konfigurátorech nenabízejí možnosti dekorace interiéru jako kompletní sady, ale jako jednotlivé části interiéru. Konfigurátor BMW například rozděluje výběr interiéru na dvě části - výběr čalounění sedadel a výběr vnitřního obložení.

2.2.9 Výběr doplňkové výbavy

Zpravidla poslední výběrovou sekcí je v konfigurátorech nabídka příplatkové volitelné výbavy. Některé automobilky sekci příplatkové výbavy rozdělují na další části - např. na balíčky výbavy (pakety) a samostatnou výbavu, nebo na výbavu a příslušenství, popř. doplňkové služby.

Téměř všechny konfigurátory upozorňují uživatele na výběr nekompatibilních položek. Konfigurátor značky Mitsubishi tuto funkci zdánlivě postrádá (obrázek 10).

2.2.10 Shrnutí konfigurace

Posledním krokem konfigurace je vždy shrnutí všech výběrů uživatele. Tato sekce konfigurátorů standardně obsahuje seznamy standardní a příplatkové výbavy, vybranou barvu, kola a interiér a výslednou cenu, případně také možnosti financování vozu. U většiny konfigurátorů rovněž nechybí alespoň základní seznam technických údajů vozidla.

Automobilky v této části konfigurace také nabízejí různé možnosti ukládání výsledných konfigurací. Způsoby ukládání konkrétních konfigurací používané analyzovanými konfigurátory jsou uvedeny v následujícím seznamu.

Obrázek 10: Výběr zdánlivě nekompatibilních položek (www.mitsubishi-motors.cz)

1. **Odkaz na výslednou konfiguraci**

   Pro konkrétní konfiguraci existuje odkaz obsahující veškeré vybrané možnosti. Po návštěvě tohoto odkazu webová aplikace obnoví z informací v odkazu konkrétní konfiguraci. Odkaz se aktualizuje při každém přidání/odebrání položky a u většiny konfigurátorů využívajících tento způsob ukládání konfigurací je dostupný také z adresního řádku prohlížeče po celou dobu konfigurace.

   Výhody: přenosnost, konfigurace je uložena i v případě mylné navigace pomocí tlačítek prohlížeče, snadné načtení konfigurace.

2. **Vygenerování kódu**

   Na konci konfigurace je uživateli nabídnut kód o určité délce. Po jeho zadání při další návštěvě uživatele webová aplikace znovu obnoví konkrétní konfiguraci.

   Výhody: pro uživatele příjemnější forma uložení kvůli délce kódu ve srovnání s délkou odkazu.

   Nevýhody: pro načtení konfigurace je nutné nejdříve vyhledat pole, do kterého je nutné kód vložit, vyžaduje externí systém (databázi) pro mapování kódů na konkrétní konfigurace.

3. **Přiřazení k zákaznickému profilu**

   Automobilka umožňuje uložení konkrétních konfigurací pro přihlášené zákazníky pomocí přiřazení výsledné konfigurace k zákaznickému profilu.

   Výhody: vyvolání podnětu k vytvoření zákaznického profilu, z konfigurací přihlášeného zákazníka je automobilkám umožněno vytvořit si o preferencích zákazníka přesnější obrázek.

   Nevýhody: pokud se jedná o jedinou možnost uložení konfigurace, tento způsob může vynucenou registrací zákazníka odradit.

4. Uložení PDF dokumentu

Zákazníkovi je nabídnuta možnost stažení dokumentu ve formátu PDF. Tento vygenerovaný dokument obsahuje seznam všech zvolených možností výsledné konfigurace, popř. další informace o vozidle. U některých automobilek je tato funkcionalita nahrazena možností zobrazení formátu vhodného pro tisk.

Výhody: možnost uložení dat obsahujících informace o konkrétní konfiguraci převoditelné do fyzické podoby.

Několik konfigurátorů také pro uložení konkrétní konfigurace využívá lokální úložiště prohlížeče. To je ale vázané ke konkrétnímu prohlížeči a zařízení, což neumožňuje přenosnost výsledné konfigurace. Jejich účelem je tedy spíše usnadnit uživateli obnovení poslední konkrétní konfigurace, kterou vytvářel. Tabulka 4 uvádí dostupnost způsobů ukládání konkrétních konfigurací u konfigurátorů jednotlivých automobilek.

Tabulka 4: Způsoby ukládání konfigurací v konfigurátorech jednotlivých automobilek

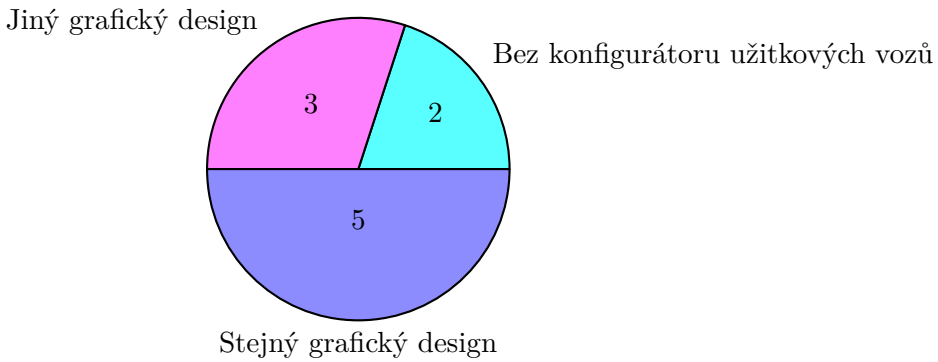
Značka	Odkaz	Vygenerování kódu	Přiřazení k profilu	Uložení PDF
Škoda	✓	✓	✓	✓
Volkswagen	✓	✓	✓	✓
Hyundai	✓			✓
Dacia	✓	✓		✓
Peugeot	✓			✓
Renault	✓	✓		✓
Toyota	✓	✓	✓	✓
Ford	✓	✓		
Kia	✓			✓
Mercedes-Benz	✓	✓		jen formát pro tisk
Seat	✓	✓	✓	✓
Citroën	✓			✓
BMW	✓		✓	✓
Opel				✓
Mazda	✓			✓
Audi	✓	✓	✓	✓
Volvo	✓	✓		✓
Mitsubishi				✓
Honda	✓			jen formát pro tisk

2.3 Konfigurátory užitkových vozů

Z analyzovaných automobilek nabízí užitkové vozy polovina z nich. Z těchto 10 automobilek všechny nabízejí konfigurátory ve formě webových aplikací pro své osobní vozy a u 8 z nich jsou také dostupné konfigurátory pro užitková vozidla.

2.3.1 Srovnání konfigurátorů užitkových a osobních vozů

Automobilky nabízející užitkové vozy je vzhledem k jejich implementaci konfigurátoru pro užitkové vozy v porovnání s implementací konfigurátoru osobních vozů automobilky možné rozdělit do 3 skupin, které ilustruje graf na obrázku 11.

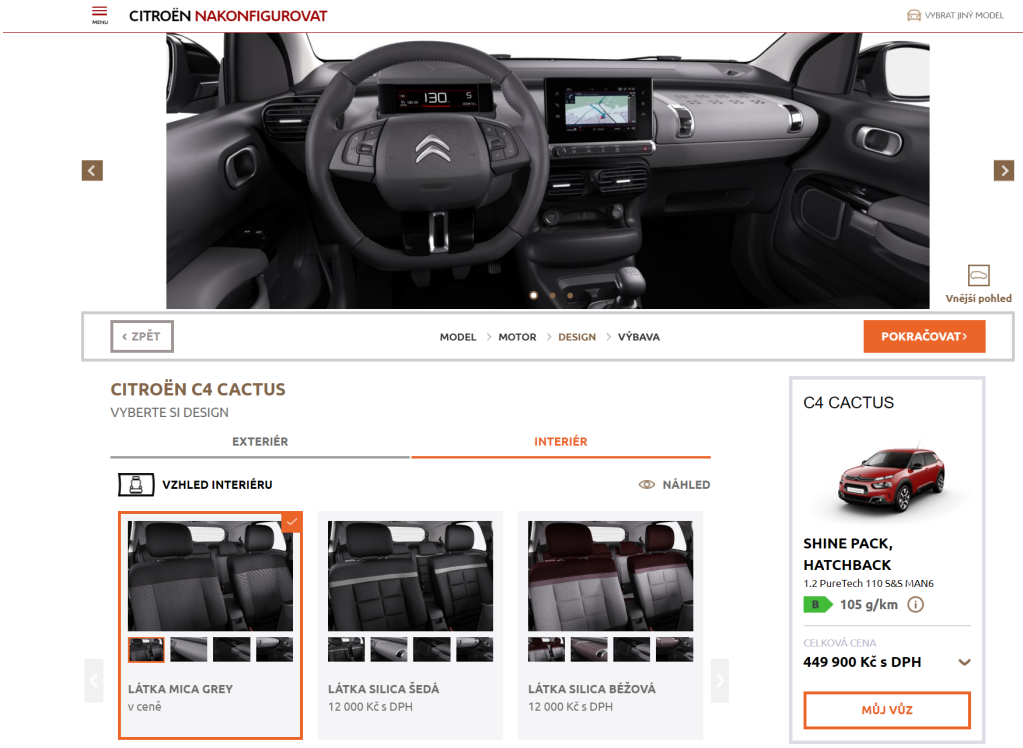


Obrázek 11: Graf způsobu implementace konfigurátorů užitkových vozů automobilkami v porovnání s konfigurátory osobních vozů

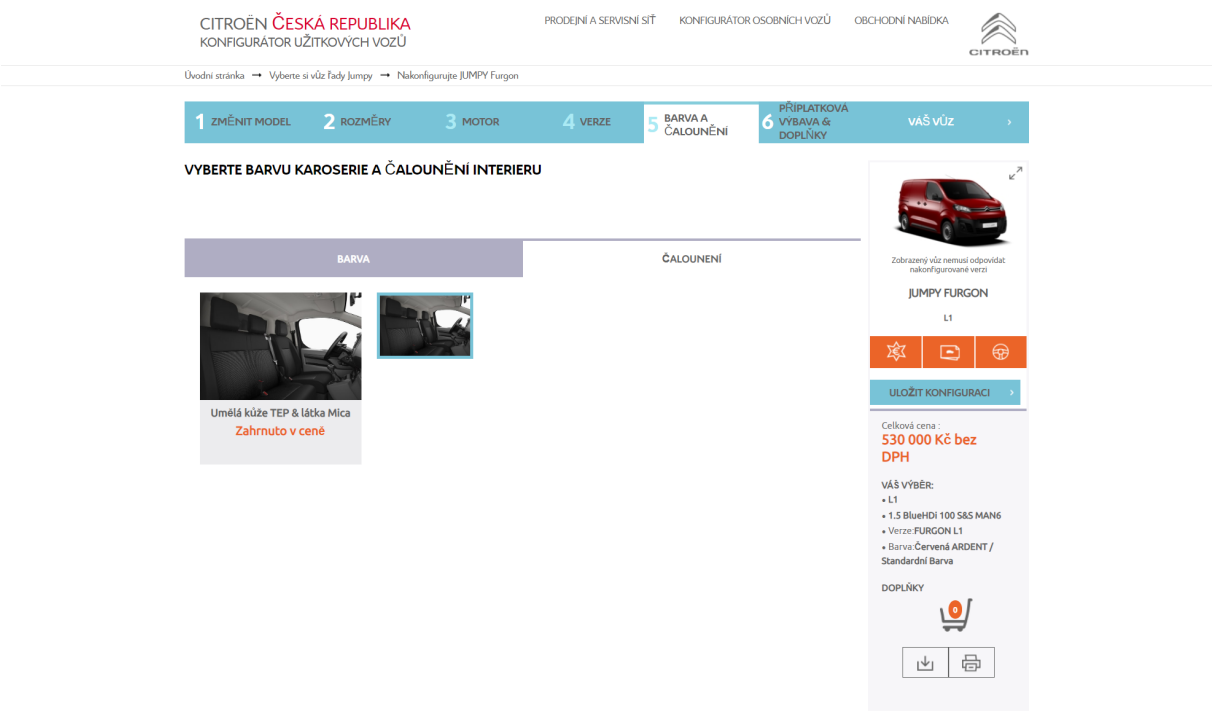
U 5 značek je grafická podoba konfigurátorů pro užitkové vozy stejná až na malé rozdíly a u zbylých 3 je grafický design podstatně rozdílný - jedná se zejména o rozdíly v rozložení prvků, jednodušší a zdánlivě méně propracovaný celkový vzhled, který může působit zastarale. Menší pozornost je také věnována vizualizaci výsledného vzhledu vozidel - náhledy podoby vozů jsou menší, dostupných je méně pohledů, elementy s výsledným vzhledem vozu nezaujímají centrální pozici na stránce. Rozdíly mezi konfigurátory značky Citroën jsou patrné na obrázcích 12a a 12b.

U konfigurátorů užitkových vozidel lze také pozorovat rozdíly ve funkčnosti a zobrazovaných datech. Kdežto u konfigurátorů osobních automobilů byl kladen důraz na zobrazování informací o spotřebě paliva a emisích vozů, u užitkových vozů jsou často tyto statistiky nahrazovány, popř. upozaďovány informacemi o objemu nákladového prostoru a počtu míst pro přepravované osoby.

Rozdíl je u některých konfigurátorů také ve stavbě jednotlivých kroků konfigurace. Standardní kroky konfigurátorů osobních vozů předchází v části konfigurátorů užitkových vozů volba provedení vozu. Jedná se zejména o výběr variant s rozdílnými velikostmi kabin a nákladových prostorů. Tento krok konfigurace je zachycen na obrázku 13.

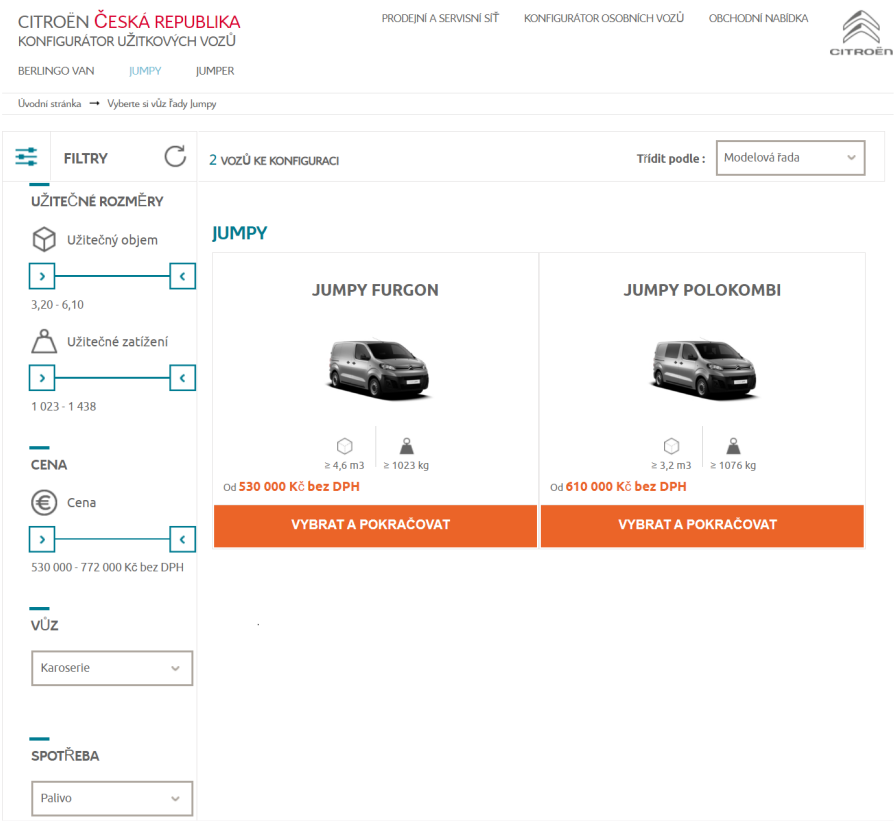


(a) konfigurátor osobních vozů



(b) konfigurátor užitkových vozů

Obrázek 12: Porovnání vzhledu konfigurátorů osobních (12a) a užitkových (12b) vozů značky Citroën (www.citroen.cz, konfigurator-business.citroen.cz)



Obrázek 13: Výběr provedení užitkového vozu Citroën Jumpy (konfigurator-business.citroen.cz)

Ze skutečností uvedených v této sekci lze usoudit, že automobilky věnují více zdrojů a úsilí budování konfigurátorů pro osobní automobily. Dostupnost konfigurátorů pro užitkové vozy pro automobilky nepředstavuje tak velkou prioritu jako dostupnost konfigurátorů pro osobní automobily.

2.4 Řešení kompatibility komponent

V analyzovaných konfigurátorech existují 2 hlavní typy způsobů ověřování kompatibility a řešení případných konfliktů podle místa, kde tyto procesy probíhají: na straně serveru nebo na straně klienta.

2.4.1 Řešení kompatibility komponent na straně serveru

Kontrola závislostí vybraných komponent probíhá na straně serveru. Jakmile se uživatel rozhodne přidat či odebrat komponentu konfigurace, vyšle se na server požadavek obsahující současnou konkrétní konfiguraci a požadovanou změnu (přidání či odebrání komponenty). Jako odpověď se pak vrací informace o tom, zdali nastane provedením takové změny konflikt, a pokud ano, nachází se v odpovědi také navrhovaná řešení tohoto konfliktu.

Zpracování závislostí a konfliktů na straně serveru využívá konfigurátor značky Škoda.

2.4.2 Příklad řešení kompatibility komponent na straně serveru - konfigurátor Škoda

Pro zobrazení varování „Upozornění - Tato položka je v rozporu s předchozími výběry“ a „Další výbava - Tato položka vyžaduje přidání dalších doplňků“, které se v konfigurátoru zobrazují u daných komponent, je od klienta vyslán požadavek obsahující informace o aktuální konkrétní konfiguraci. Odpovědí na tento požadavek jsou seznamy položek, pro které jsou tato varování platná.

Poté, co se uživatel rozhodne pro přidání či odebrání položky, provede konfigurátor validaci změn. V požadavku se nachází aktuální konfigurace a požadovaná změna konfigurace. V odpovědi se pak nachází informace o tom, zdali změna vyvolá konflikt. Pokud ano, nachází se v odpovědi na požadavek také řešení daného konfliktu, což vyvolá zobrazení informačního dialogu uživateli, který se pak může rozhodnout o postupu. Pokud ke konfliktu nedošlo, provede se požadovaná změna konfigurace. Část přenášených dat požadavku ve formátu JSON je uvedena ve výpisu 1, odpověď pak ve výpisu 2.

```
{
  "configuration": {
    "snapshotVersion": "687fd4f8-de74-415a-b7b6-cb105f46be33",
    "modelId": "CZE;skoda;2020;3V3445;2;GYOLYOL;mda20200320024041;cs-CZ
       ;;69000;69000",
    "trimlineId": "3V4|Style6900069000",
    "color": "9799",
    "interior": "LA",
    "extraEquipments": [
      "MLAKFB1"
    ]
  },
  "change": {
    "property": "extraEquipments",
    "quantity": 1,
    "id": "GPNFPNF"
  },
  "numberOfSolutions": 1,
  "requestId": "_1587232916410"
}
```

Výpis 1: Data přenášená v požadavku na odebrání komponenty v konfigurátoru Škoda

```
{
  "isConflict": true,
  "configurations": [
    {
      "modelId": "CZE;skoda;2020;3V3445;2;GYOLYOL;mda20200320024041;cs-CZ
          ;;69000;69000",
      "trimlineId": "3V4|Style6900069000",
      "color": "9799",
      "wheel": null,
      "interior": "LA",
      "extraEquipments": [
        "MSON5XH",
        "GPNFPNF",
        "MEIH5MT",
        "MLAKFB1"
      ],
      "snapshotVersion": "687fd4f8-de74-415a-b7b6-cb105f46be33",
      "price": {
        "price": -842400
      },
      "priceDifference": "-842400.0"
    }
  ],
  "requestId": "_1587232916410",
  "manualAdd": [
    "MEIH5MT",
    "MEIH5TB"
  ],
  "manualRemove": [],
  "isOverweight": false
}
```


Výpis 2: Data přenášená v odpovědi na požadavek odebrání komponenty v konfigurátoru Škoda

2.4.3 Řešení kompatibility komponent na straně klienta

Druhým způsobem ověřování kompatibility komponent a celkového procesu budování konkrétních konfigurací je zpracovávání dat týkajících se omezení komponent na straně klienta. Informace o závislostech komponent jsou přeneseny na stranu klienta a k detekci a řešení konfliktů již není nutná komunikace se serverem.

Tohoto řešení využívají například konfigurátory Volkswagen a BMW.

2.4.4 Příklad řešení kompatibility komponent na straně serveru - konfigurátor Volkswagen

Přenášená data komponent, která mají specifické požadavky na přítomnost nebo naopak nepřítomnost dalších komponent, obsahují objekt *regeln*. Tento objekt může obsahovat jednu či více z následujících polí hodnot.

ZWANG - v tomto poli se nachází kódy komponent, které musí být přítomny v konfiguraci vozidla pro úspěšné přidání původní komponenty.

VERBOT - obsahuje kódy komponent, které nemohou být přítomny v konfiguraci společně s původní komponentou.

ALTZWANGx - x je nahrazeno konkrétním číslem, jedná se o pole kódů, mezi kterými je umožněno uživateli provést výběr. Pro úspěšné přidání původní komponenty je nutné, aby byla v konfiguraci přítomna vždy alespoň jedna z komponent uvedených v jednotlivých *ALTZWANG* polích.

Uživateli jsou zobrazovány dialogy opět pouze tehdy, pokud jeho výběr vyžaduje přidání či odebrání jiné než původně vybrané komponenty, popř. pokud je po uživateli vyžadována volba mezi alternativními komponentami.

2.4.5 Příklad řešení kompatibility komponent na straně serveru - konfigurátor BMW

Konfigurátor značky BMW také využívá způsob řešení závislostí komponent na straně klienta a data komponent přenesených na stranu klienta obsahují informace o závislostech uložené ve struktuře (výpis 3) podobné té, kterou využívá automobilka Volkswagen.

Pole *causesRemovals* obsahuje kódy komponent, se kterými nemůže existovat daná komponenta v jedné konkrétní konfiguraci.

Pole polí *causesAdditions* naopak určuje, které komponenty jsou nutné pro přidání původní komponenty. Pro úspěšné přidání komponenty do konkrétní konfigurace musí taková konfigurace obsahovat vždy alespoň jednu komponentu z komponent identifikovaných kódy v jednotlivých polích umístěných uvnitř hlavního pole *causesAdditions*.

Uvedený příklad datové struktury tedy určuje, že pro přidání takové komponenty nesmí konfigurace obsahovat komponentu s kódem *S03AP*, a naopak musí obsahovat komponentu označenou kódem *S06WA* a také jednu z komponent označených kódy *S0544* a *S05DF*.

```
"causesRemovals": [
   "S03AP"
],
"causesAdditions":[
   [
      "S0544",
      "S05DF"
   ],
   [
      "S06WA"
   ]
]
```

Výpis 3: Část dat komponenty z konfigurátoru BMW se závislostmi na ostatních komponentách

2.4.6 Srovnání principů řešení kompatibility komponent

Výhody řešení závislostí na straně serveru:

- velikost přenášených dat po zobrazení stránky výběru výbavy je snížena o objem dat nutných k určení závislostí mezi komponentami,
- automobilka neodkrývá veškerá data o závislostech komponent straně klienta.

Výhody řešení závislostí na straně klienta:

- není nutné server zatěžovat samostatnými požadavky pro každou změnu konfigurace obsahující komponenty se závislostmi,
- server nemusí provádět samotný výpočet řešení konfliktu závislostí.

Konfigurátory, které využívají řešení závislostí na straně serveru neumožňují jednoduché získávání informací o závislostech komponent skrze svá API. Získávání závislostí skrze tato API by vyžadovalo vysílat jednotlivé požadavky pro všechny komponenty vozidel společně se všemi možnými kombinacemi těchto komponent v konfigurátoru. Díky extrémně vysokým počtům požadavků nutných k získání kompletních informací o závislostech komponent pro každý model automobilu by bylo toto řešení nepraktické.

Praktičtější možností by v takových případech byla možnost převzetí závislostí z ceníku automobilu ve formátu PDF, pokud jsou závislosti v ceníku dané automobilky uvedeny. Další možností je manuální doplňování nekompatibilních komponent (buďto díky testování některých kombinací nebo podle jasného vyplynutí z názvu komponent). Poslední možností je vynechání zachycování údajů o závislostech komponent a upozornění uživatele konfigurátoru na skutečnost, že automatická kontrola kompatibility komponent konfigurace je pro danou automobilku nedostupná.

3 Implementace konfiguračního systému

Tato implementační část práce je také doprovázena teoretickou příručkou popisující vhodné volby technologií při budování konfiguračních systémů (příloha C).

3.1 Základní struktura konfiguračního systému

Pro implementaci kompletního konfiguračního systému automobilů je nutný návrh několika modulů (obrázek 14), které jsou, pokud možno, na sobě co nejvíce nezávislé kvůli zajištění ulehčení případných budoucích úprav či kompletních výměn těchto modulů. [9]

1. Nástroj na stahování dat vozidel - cílem programu je zautomatizování získávání co nejkompletnějších dat o vozidlech a jejich konfiguračních možnostech.
2. Databáze - nutná pro poskytování perzistentního úložiště výstupních dat po jejich stažení výše uvedeným nástrojem.
3. API - webová služba pro umožnění přenosu konfiguračních dat z databáze na stranu uživatele. Na základě srovnání principů řešení závislostí (sekce 2.4.6) byl pro implementaci konfiguračního systému vybrán princip ověřování kompatibility komponent na straně klienta, což umožňuje jednodušší budoucí rozšíření aplikace díky menší zátěži vyvíjené uživateli.
4. Webový server - účelem je umožnit uživatelům přístup k webovým stránkám a webové aplikaci plněním dotazů odeslaných uživateli na server.
5. Webová aplikace - aplikace zajišťující uživatelsky přívětivé grafické rozhraní a logiku samotné konfigurace vozidel.

3.2 Stahování informací o konfiguracích od automobilek

Nástroj, který umožňuje stahování informací o vozidlech přímo od automobilek musí obsahovat několik vrstev rozdělených podle funkce, kterou daná vrstva plní.

1. **Vrstva stahování dat**

   Nejdříve musí být stažena samotná data. K těmto datům je možno přistupovat pomocí HTTP požadavků, na které odpovídají API automobilek odesláním dat.

2. **Vrstva konverze dat**

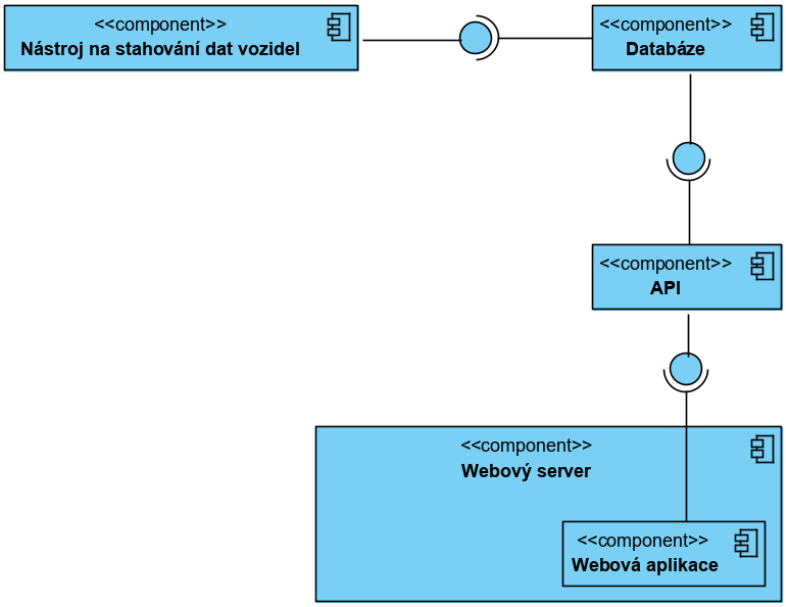
   Jelikož jsou struktury dat o vozidlech skládány tak, aby přesně vyhovovaly požadavkům a možnostem automobilek, musí být součástí programu vrstva, která se stará o spojování, zpracování a standardizaci dat získaných z odpovědí na vyslané požadavky. Z toho také vyplývá, že pro získávání dat od automobilek používajících různou strukturu dat, musí nástroj mít zvláštní logiku unifikace těchto dat.

Výstupem této vrstvy jsou objekty se standardizovanou strukturou představující jednotlivé modely, konkrétní automobily a konfigurační možnosti. Tento způsob prozatímního uchovávání konfiguračních dat různých automobilek ve společných objektech umožňuje velkou flexibilitu další práce s těmito daty - kromě jejich ukládání do perzistentní podoby pro účely konfigurace se například nabízí jejich využití při počítání statistik jednotlivých automobilek či modelů.

3. **Vrstva ukládání dat**

   Poslední vrstva se stará o export objektů vytvořených ve 2. vrstvě do souboru či do databáze pro budoucí využití.

Pro implementaci tohoto nástroje byl vybrán programovací jazyk Java zejména díky univerzálnosti uplatnění jazyka podporovaného mimo jiné také širokým výběrem dostupných knihoven.



Obrázek 14: Diagram komponent zachycující základní strukturu konfiguračního systému

3.2.1 Objekt uchovávající data o modelech

Objekty uchovávající data o modelech musí být nezávislé struktury obsahující veškeré informace nutné k identifikaci modelu a k jeho konfiguraci. Dále tato struktura musí obsahovat technické údaje vozidla, které budou v procesu konfigurace užívány k informování uživatele o různých technických aspektech vozu.

Pro vytvoření báze této struktury lze využít konfiguračního modelu s typy komponent a základními omezeními.

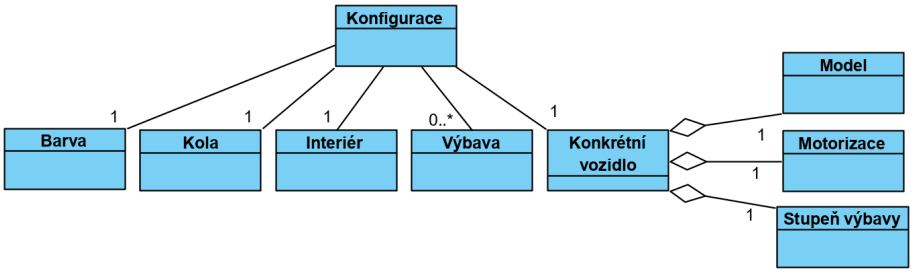
3.2.1.1 Konfigurační model

Konfigurační model pro konfiguraci konkrétních automobilů musí být:

- dostatečně výstižný, aby zahrnoval veškeré možné případy konfigurací, tzn. musí sjednotit rozdíly v konfiguracích mezi různými automobilkami a typy vozidel,
- co nejvíce stručný, aby se model nenafukoval kvůli rozdílům konfigurací vozidel mezi jednotlivými automobilkami a typy vozidel.

Z diagramu základního konfiguračního modelu (obrázek 15) lze vidět, že konfigurace obsahuje vždy 1 instanci barvy, kol, interiéru a konkrétního vozidla, které je identifikováno zvolenými možnostmi - modelem, motorizací a stupněm výbavy. Součástí konfigurace je také neurčitý počet položek volitelné výbavy, do které spadá fyzická výbava vozidla, služby i dodatečné příslušenství.

Větší počet barev (některé automobilky nabízejí vozidla v kombinaci barev) lze vyřešit pomocí atributu určujícího typ barvy u jednotlivých instancí barev. Podobného řešení lze využít k odlišení fyzické výbavy od služeb - opět je využito atributu instance označující kategorii výbavy.



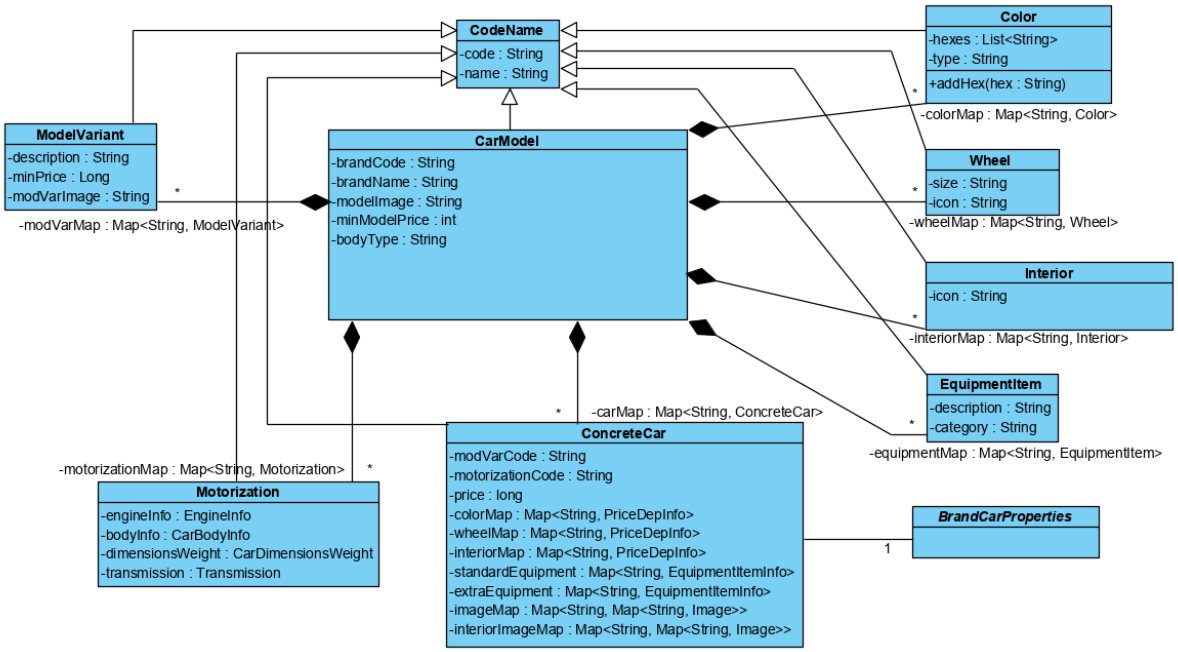
Obrázek 15: Diagram základního konfiguračního modelu

3.2.1.2 Struktura objektu uchovávajícího data o modelech

Struktura objektu, jehož účelem je uchovávání dat o modelech, vychází ze základního konfiguračního modelu. V serializované podobě budou data uchovávána ve formátu JSON. Formát JSON nabízí mnoho výhod - formát je široce rozšířený a v ohledu objemu serializovaných dat velmi úsporný. Využití tohoto formátu pro přenos dat podporuje také jeho velmi rychlá serializace a deserializace. [9][10][11]

Na obrázku 16[1] lze vidět strukturu uchovávající data modelu. Diagram popisuje strukturu tříd v Javě, výsledná struktura ve formátu JSON tuto strukturu kopíruje až na třídu *BrandCarProperties*, která je bázovou třídou pro třídy vázané ke specifickým automobilkám. Účelem takových tříd je udržování dat po dobu zpracování původních dat automobilek, avšak po ukončení zpracování jsou již tato data nepotřebná.

[1]Diagramy v této práci obsahují kvůli úspoře místa a jasnosti zobrazení pouze části, které jsou pro účely znázornění konkrétních jevů relevantní. Kvůli úspoře místa jsou také některé kolekce zakresleny přímo mezi atributy tříd.



Obrázek 16: Diagram struktury tříd uchovávající data o modelu

Objekty představující barvy, kola, interiéry, položky výbavy, stupně výbavy (modelové varianty) a motorizace (pro zjednodušení jsou souhrnně tyto objekty nazývány komponentami) jsou ukládány do map s pomocí jejich identifikačních kódů. Tyto kódy jsou většinou převzaty z původních dat automobilek, v případě jejich nedostupnosti v originálních datech jsou tyto kódy generovány (z názvů, popř. hodnot atributů jednotlivých položek). Třídy komponent proto také dědí ze třídy *CodeName*, která uchovává kódy a názvy jednotlivých komponent.

Informace o barvách, kolech, interiérech a výbavě jsou rozděleny na více částí - obecné informace komponent, které nejsou vázané na specifický stupeň výbavy či motorizaci (např. názvy a popisy komponent) jsou ukládány do patřičných map společně pro všechny konkrétní vozy modelu.

Informace vázané na konkrétní vozy (tedy kombinace modelu, motorizace a stupně výbavy) jsou ukládány do map instancí třídy *ConcreteCar*, která tyto konkrétní vozy představuje. Záměrem rozdělení uchovávaných informací o komponentách na více částí je omezení opakování informací a tím i úspora celkového objemu, který výsledná data zabírají.

Ukládání barev Pro ukládání dostupných barev a jejich vizualizaci byla kvůli co nejlepší standardizaci různých způsobů vizualizace barevných odstínů vybrána metoda ukládání kódů barev. Kvůli existenci kombinací barev v konfigurátorech některých automobilek je možné do struktury uložit více kódů barev. S rozdělením barev na kategorie napomáhá atribut obsahující standardizovaný kód typu barvy.

Ukládání kol Kola jsou ukládána společně s odkazem na ikonu kola a také jejich velikostí (udávanou v palcích). Velikost kol je ideálním kritériem pro rozdělení kol na kategorie kvůli mnohem lepší dostupnosti těchto dat oproti informacím o materiálu, ze kterých byla kola vyrobena.

Ukládání interiérů Kromě standardního názvu a kódu interiéru obsahují objekty reprezentující možnost výběru interiéru vozu také odkaz na ikonu interiéru. Objekty neobsahují informaci o typu interiéru z několika důvodů:

- rozdělení interiérových možností na typy není mezi konfigurátory automobilek běžné,
- většina automobilek nabízí výběr z poměrně malého počtu dostupných interiérů a jejich rozdělení na kategorie tak není nutné.

Ukládání položek výbavy Položky výbavy obsahují popis výbavy převzatý od automobilky a standardizovaný název kategorie, do které se výbava řadí (více informací o standardizaci kategorií se nachází v sekci 3.2.2.3).

Ukládání stupňů výbavy V objektech, které představují jednotlivé stupně výbavy, se nachází odkaz na obrázek reprezentující danou úroveň výbavy, počáteční cenu vozů s touto výbavou a popis stupně výbavy, který je uložen ve formátu, který podporuje ukládání textu s odrážkami.

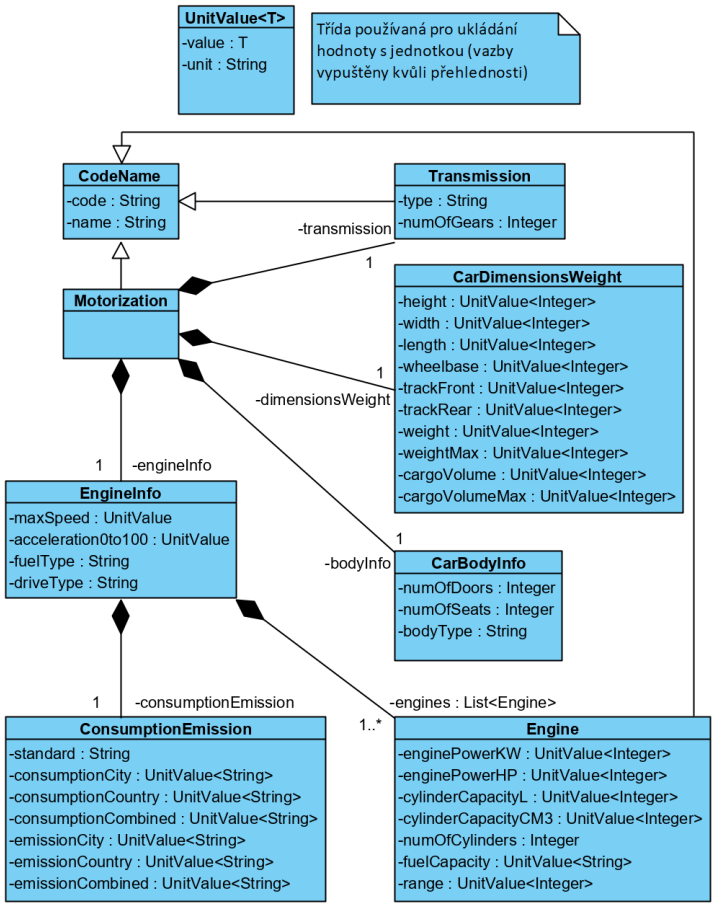
Ukládání motorizací Informace o motorizacích jsou ukládány do struktury viditelné na obrázku 17. Motorizace je tedy rozdělena na několik částí obsahujících technické údaje o motoru, převodovce, rozměrech a váhách vozu a o provedení karoserie.

Objekt obsahující informace o motoru je dále rozdělen - na objekt s informacemi o spotřebě a emisích (včetně informace o standardu použitém při měření těchto hodnot) a seznam obsahující jeden či více objektů reprezentující jednotlivé motory. Strukturu dovolující ukládání informací o více motorech je možné využít zejména u automobilů s hybridním pohonem či u elektromobilů s několika elektromotory.

Ne všechny uvedené technické údaje je vždy možné získat z dat získaných od automobilek. Je proto nutné umožnit jejich doplňování, popř. počítat s možností, že ne všechny údaje budou kompletně vyplněny (více informací o doplňování technických údajů je uvedeno v sekci 3.2.2.1).

Dále je nutno uvést, že technické údaje jsou pouze orientační díky tomu, že nejsou ovlivněny váhou a dalšími charakteristikami plynoucími z výběru různých typů kol a další výbavy.

Motorizace, stupně výbavy či modely mohou také ve zvláštních případech zastupovat chybějící sekci výběru u atypicky rozdělených konfigurátorů automobilek. Příkladem může být konfigurátor automobilky Ford, jehož součástí je výběr karoserie. Automobily Ford stačí rozdělit do samostatných modelů podle jednotlivých možností výběrů karoserie. U výběru provedení užitkových vozů některých automobilek je opět možné tuto volbu převést např. na volbu motorizace.



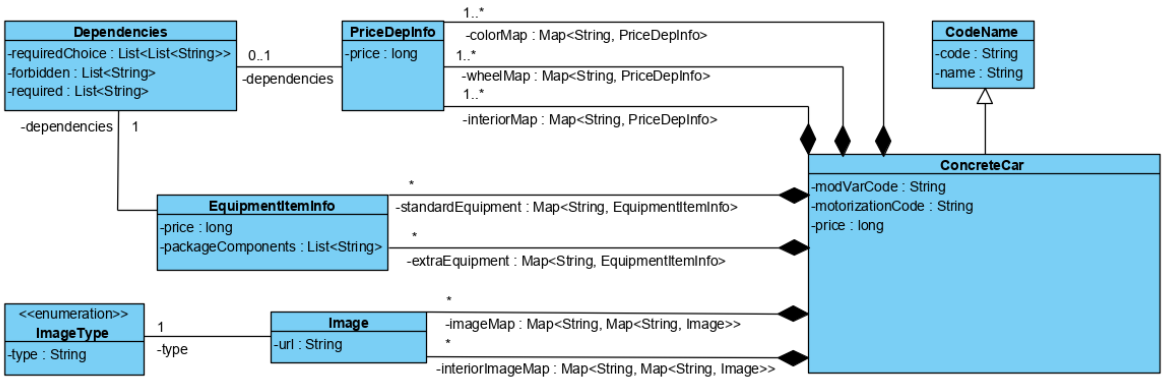
Obrázek 17: Diagram struktury tříd obsahujících údaje o motorizaci

Ukládání konkrétních vozů Strukturu určenou pro ukládání konkrétních provedení vozů je možno vidět na obrázku 18. Díky kódům motorizace a stupně výbavy je možné přiřadit k vozu konkrétní motorizaci a stupeň výbavy.

Mapy barev, kol a interiérů používají jako klíče kódy komponent a hodnotami jsou objekty s cenou a závislostmi těchto komponent. Třída *Dependencies* určená k reprezentaci závislostí mezi komponentami obsahuje pro tento účel 3 seznamy. Seznam s názvem *required* obsahuje kódy komponent, které je nutné pro přidání komponenty obsahující tato omezení také přidat. Seznam *forbidden* naopak obsahuje kódy komponent, se kterými není daná komponenta kompatibilní. Poslední seznam se jménem *requiredChoice* obsahuje další seznamy kódů komponent. Pro úspěšné přidání komponenty obsahující tato omezení je nutná přítomnost vždy alespoň jedné z komponent v konkrétní konfiguraci z každého zanořeného seznamu kódů komponent.

Struktura obsahuje celkem 2 instance map obsahujících doplňující informace týkající se výbavy vozu. První z nich je *standardEquipment*. Výbava patřící do této mapy je součástí standardní výbavy vozu - znamená to tedy, že tato výbava je u vozu neměnná (vůz s touto výbavou začíná a není ji možné odstranit či s ní jinak manipulovat). Druhou sadou výbavy je volitelná výbava, informace týkající se této výbavy jsou uloženy v mapě *extraEquipment*. Jak název na-

povídá, uživateli je umožněno vybírat si z této výbavy položky.



Obrázek 18: Diagram popisující strukturu tříd reprezentující konkrétní vozy

Třída *EquipmentItemInfo* obsahuje navíc oproti třídě *PriceDepInfo* seznam *packageComponents*. Pokud obsahuje tento seznam kódy komponent, znamená to, že je komponenta obsahující tento seznam paketem - balíčkem výbavy, který zahrnuje veškerou výbavu označenou kódy v seznamu a ceny těchto jednotlivých položek výbavy jsou již zahrnuty v ceně paketu.

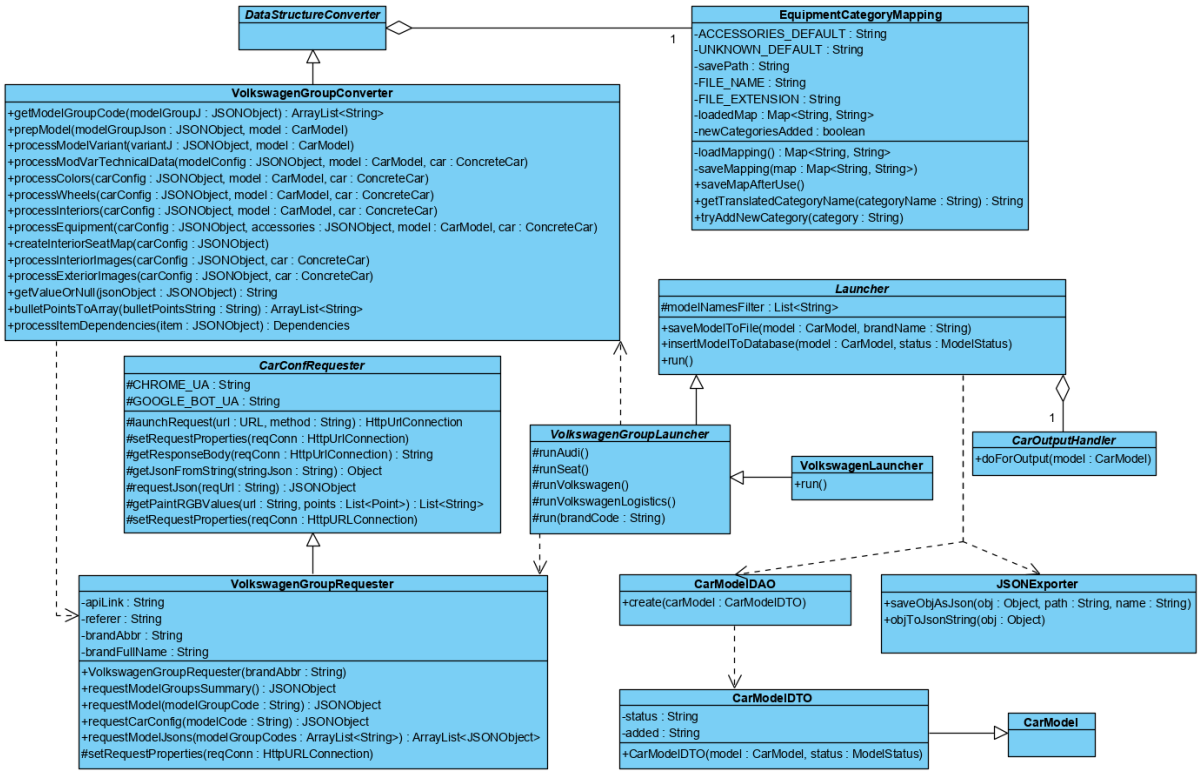
Reprezentace konkrétního vozu také obsahuje objekty *Image* uložené v mapách *imageMap* a *interiorImageMap*. Vzhledem k rozmanitosti způsobů generace odkazů na obrázky obsahující pohledy na exteriér a interiér automobilů napříč různými konfigurátory jsou ukládány do struktury přímo odkazy na tyto obrázky.

Klíčem map obrázků exteriéru vozů je kombinace kódu barvy a kol, u mapy s obrázky interiéru vozu je klíčem pouze kód interiéru. Hodnotou obou těchto map jsou pak další mapy, kde je klíčem kód typu pohledu a hodnotou již samotný objekt *Image*. Díky této implementaci ukládání obrázků odrážejí obrázky nejzásadnější rozdíly ve vzhledu vozů plynoucí z výběru motorizace, stupně výbavy, barvy, kol a interiéru, nemohou však už odrážet další změny ve vzhledu způsobené výběrem volitelných položek výbavy (v případě, že daná automobilka tyto změny vizualizace výsledné podoby vozu v závislosti na vybrané volitelné výbavě podporuje).

Kvůli standardizaci počtu a druhů ukládaných pohledů napříč automobilkami a kvůli snížení velikosti výsledné struktury byly vybrány 3 pohledy na exteriér a 3 pohledy na interiér, jejichž odkazy budou nástrojem ukládány. Jedná se o přední pohled, pohled z boku a pohled zezadu, a to jak pro exteriér, tak interiér vozu. Tyto počty a typy pohledů byly zvoleny s ohledem na počty obrázků zachycující výslednou podobu vozu u jednotlivých automobilek v tabulce 2. Jelikož ne všechny automobilky ve svých konfigurátorech nabízejí 3 pohledy na exteriér a 3 pohledy na interiér, je nutné při pozdějším využití těchto struktur počítat s tím, že některé typy pohledů nebudou vždy k dispozici.

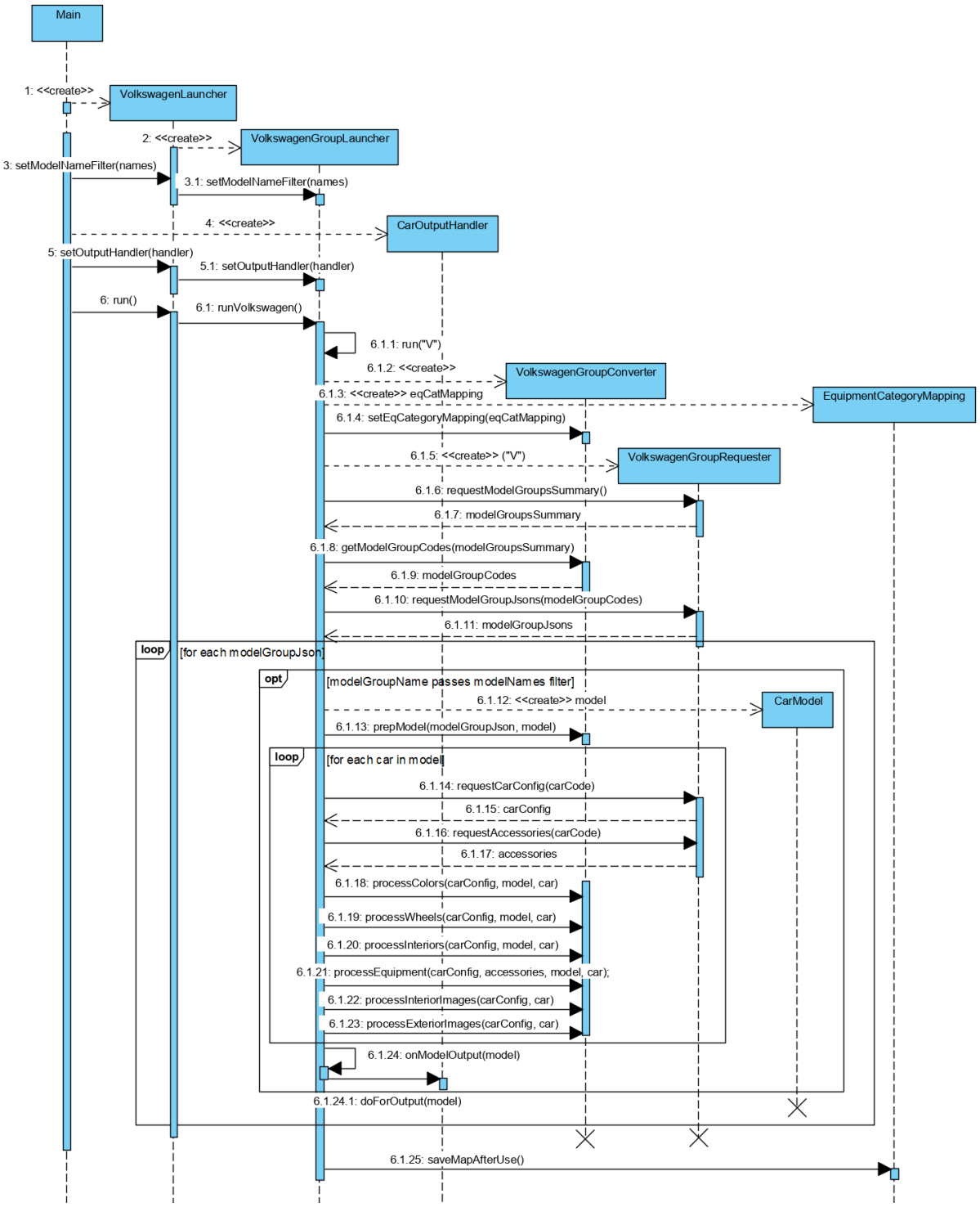
3.2.2 Průběh a způsob zpracování dat

Struktura hlavních tříd starajících se o zpracování dat je uvedena v třídním diagramu na obrázku 19. Nástroj je spouštěn pomocí konkrétních implementací abstraktní třídy *Launcher*. Vrstvu zodpovědnou za stahování dat reprezentují konkrétní implementace abstraktní třídy *CarConfRequester*. O samotnou konverzi dat z původní struktury automobilky se starají implementace abstraktní třídy *DataStructureConverter*. Uložení dat do databáze a do souboru řeší třídy *CarModelDAO* a *JSONExporter*. Průběh zpracování dat automobilky Volkswagen popisuje sekvenční diagram 20.



Obrázek 19: Diagram tříd obstarávajících zpracování dat automobilek

Nastavení programu je znázorněno ve výpisu kódu 4. Na začátku procesu je vytvořen objekt konkrétní implementace třídy *Launcher*. U automobilek Volkswagen, Seat a Audi, jejichž struktura konfiguračních dat se liší spíše jen v detailech, dědí ze třídy Launcher ještě navíc třída VolkswagenGroupLauncher. Zpracování konfiguračních dat zmíněných automobilek je spouštěno přes potomky třídy VolkswagenGroupLauncher (v ukázce kódu je touto třídou VolkswagenLauncher). Objektu třídy spouštějící zpracování dat je možno předat seznam jmen modelů, pro které je nutno stáhnout a zpracovat data. Pokud objektu není předán seznam jmen modelů, jsou zpracovány všechny modely automobilky.



Obrázek 20: Sekvenční diagram popisující zpracování konfiguračních dat automobilky Volkswagen

Anonymní vnitřní třída s překrytou metodou *doForOutput* určuje, co se stane se zpracovanou strukturou modelu. V ukázce je model uložen do souboru a vložen do databáze. Serializace a deserializace objektové struktury v programu probíhá s pomocí knihovny Jackson, která proces serializace a deserializace objektů automatizuje.

```
Launcher launcher = new VolkswagenLauncher();
List<String> modelNamesFilter = new ArrayList<>(Arrays.asList("Polo", "Passat",
    "Golf"));
launcher.setModelNameFilter(modelNamesFilter);
launcher.setOutputHandler(new CarOutputHandler()
{
   @Override
   public void doForOutput(CarModel model)
   {
      launcher.saveModelToFile(model, model.getBrandName());
      launcher.insertModelToDatabase(model, ModelStatus.ACTIVE);
   }
});
launcher.run();
```

Výpis 4: Nastavení spuštění nástroje pro zpracování dat automobilek

Po zavolání metody spouštějící proces zpracování dat (v ukázce metoda *run*) jsou stažena potřebná data z API vybrané automobilky. Tato data jsou poté postupně zpracována po jednotlivých modelech automobilky. Pro každý z modelů je vytvořen objekt třídy *CarModel*, který seskupuje veškeré informace o konkrétním modelu. Tento objekt je dále doplněn informacemi o stupních výbavy (reprezentovány třídou *ModelVariant*) a motorizacemi. Do unifikované podoby jsou také převedena data reprezentující dostupné možnosti výběru barev, kol, interiérů a výbavy a dále také odkazy na jednotlivé obrázky s pohledy na exteriér a interiér vozu.

Příklady výsledných podob unifikovaných konfiguračních dat jsou v příloze D.3. Důležité kroky zpracování dat od automobilek jsou níže uvedeny detailněji.

3.2.2.1 Zpracování informací o motorizacích

Vzhledem k velkým rozdílům počtu a druhů dostupných technických údajů vztahujících se k motorizacím u jednotlivých automobilek byl implementován systém ukládání a načítání dat motorizací z a do souboru.

Při prvním zpracování dat motorizací dostupných díky API automobilky jsou objekty uchovávající tato data ukládány do souborů. Tyto soubory pak mohou být manuálně doplněny či upraveny a při dalším zpracování modelu, u kterého jsou tyto motorizace dostupné, jsou k němu přiřazeny objekty *Motorization* vytvořené pomocí deserializace dat načtených z jednotlivých souborů obsahujících data o motorizacích.

3.2.2.2 Zpracování možností barev vozů

U automobilek, které používají pro vizualizaci barevných odstínů ve svých konfigurátorech ikony, nejsou hodnoty barevných odstínů vždy dostupné v konfiguračních datech. Nástroj pro zpracování dat automobilek proto k získání hodnoty barvy využívá dostupných odkazů ikon získaných z API automobilek, kde je barevná hodnota vybraných pixelů těchto ikon ukládána ve formě hexadecimálních RGB kódů.

3.2.2.3 Standardizace typů komponent a technických údajů

Ke standardizaci typů barev, kol, interiérů a některých technických údajů, jako jsou např. typ pohonu, převodovky atd., je využíván systém provázání specifických kódů typů zmíněných komponent a údajů jednotlivých automobilek se standardizovanými kódy.

K tomuto účelu existuje v nástroji několik mapových struktur, výčty a funkce, které vrací standardizovanou verzi kódu předaného parametrem. Pokud v mapových strukturách neexistuje k specifickému kódu automobilky standardizovaná verze tohoto kódu, je vyvolána výjimka upozorňující na tuto skutečnost.

Speciální přístup je věnován početnější skupině překládaných kódů představující kategorie výbavy, jejichž standardizace je nutná pro přehlednější porovnání konfigurací modelů různých automobilek.

Před zpracováním dat modelů jsou do proměnné v instanci třídy EquipmentCategoryMapping ze souboru načteny dvojice hodnot představující názvy (popř. kódy) kategorií výbav automobilek a standardizované názvy kategorií výbav. Při zpracování dat představujících položky výbavy jsou do zmíněné proměnné vkládány nové názvy kategorií a vytvářeným standardizovaným objektům reprezentující položky výbavy jsou přiřazovány standardizované názvy kategorií.

Aktualizovaná paměťová reprezentace původních a standardizovaných kategorií je na konci procesu zpracování dat modelů zpátky serializována do souboru, ve kterém je možno provádět nastavení překladu názvů kategorií.

3.2.3 Výsledky použití nástroje pro zpracování konfiguračních dat automobilek

Implementovaný nástroj dokáže získat a následně převést do navrženého standardizovaného formátu konfigurační data 3 automobilek. Počty zpracovaných modelů jsou následující:

- 28 modelů osobních vozů značky Audi,
- 5 modelů osobních vozů značky Seat,
- 14 modelů osobních vozů značky Volkswagen,
- 13 modelů užitkových vozů značky Volkswagen.

Celkem je tedy nástroj schopný zpracovat 60 modelů automobilů 3 zmíněných automobilek. Nástroj je také připraven bez jakýchkoli změn zpracovat případné další modely těchto značek, jakmile budou zpřístupněny v jejich konfigurátorech.

3.3 Perzistentní ukládání konfiguračních dat

3.3.1 Výběr vhodného databázového systému

Pro ukládání konfiguračních dat a jejich následné čtení za účelem přenosu těchto dat do webové aplikace je velmi důležitý správný výběr databázového systému, který může zásadně ovlivnit celkový výkon aplikace a možnosti budoucí rozšiřitelnosti konfiguračního sytému.

Výběr databázového systému závisí na funkcích, které bude tento systém plnit. Potřeba zapisovat nová data (popř. aktualizovat ta stávající) bude vzhledem k počtu požadavků na čtení těchto dat jen velmi malá a nepříliš častá.

V kontextu konfiguračního systému bude tedy velmi důležitým kritériem výběru databázového systému rychlost čtení a výběru dat. Velkou roli naopak při výběru databáze nehraje rychlost zápisu dat ani podpora transakcí.

Ve srovnání s relačními databázovými systémy nabízí dokumentově orientované databáze:

- lepší škálovatelnost,
- nižší odezvu,
- větší jednoduchost návrhu a údržby. [12]

Databázové systémy uzpůsobené k ukládání dokumentů také mohou nabídnout větší výkon při čtení dat oproti relačním databázovým systémům. Ve srovnání rychlosti [13] mezi relačním databázovým systémem PostgreSQL a NoSQL databázovým systémem MongoDB v simulaci čtení dat velkým počtem uživatelů jednoznačně zvítězil systém MongoDB, jehož čas čtení dat byl v mnohých případech více než desetinásobně menší než u PostgreSQL.

Z udaných důvodů byl pro implementaci konfiguračního systému zvolen dokumentově orientovaný databázový systém. Z dostupných dokumentově orientovaných databázových systémů byl vybrán systém MongoDB díky dostupnosti licence, možnostem škálování systému a možnostem agregace dat.

3.3.2 Požadavky na výstupy databázového systému

Ve webové aplikaci bude nutné v jednotlivých krocích konfigurace využití různých dat. Díky omezení přenášených dat lze přenášet z databáze pouze data relevantní k aktuálnímu stavu konfigurace, čímž se redukuje objem přenášených dat a zmenšuje se tak celková doba přenosu.

Konfiguraci automobilů lze rozdělit do těchto kroků:

1. výběr automobilky,
2. výběr modelu,
3. výběr stupně výbavy a motorizace,
4. výběr barvy, kol, interiéru a dalších komponent výbavy.

Níže jsou uvedena data potřebná k implementaci jednotlivých kroků konfigurace. Návrhy vzhledu a funkcionality jednotlivých kroků jsou uvedeny v sekci 3.6.1.

Výběr automobilky K výběru automobilky stačí seznam kódů a názvů dostupných automobilek. Pro lepší informovanost uživatele lze také přidat počet dostupných modelů jednotlivých automobilek.

Výběr modelu Pro výběr modelu bude potřeba zajistit přístup ke kódům a názvům modelů automobilky, jejich základním cenám a k odkazu na obrázky těchto modelů. U výběru modelu by také neměl chybět filtr omezující výběr podle typu karoserie modelů, a proto je nutné přidat tuto informaci k těm již zmíněným.

Výběr stupně výbavy a motorizace Samozřejmostí je mít k dispozici informace o stupních výbavy a dostupných motorizacích modelu, které kromě kódu a názvu motorizace obsahují také technické údaje, podle kterých je možno zavést ve webové aplikaci možnost omezení výběru dostupných motorizací. Jednotlivé stupně výbavy by měly obsahovat informace o výbavě, kterou tyto úrovně přinášejí, a také odkaz na obrázek znázorňující podobu vozidla s daným stupněm výbavy.

Výběr stupně výbavy a motorizace identifikuje konkrétní vozidlo, které bude konfigurováno uživatelem. Je tedy nutné mít k dispozici základní informace o konkrétních vozech - název a cenu, a kromě kódu konkrétního vozu také kódy motorizace a stupně výbavy, ke kterému se vozidlo řadí.

Výběr barvy, kol, interiéru a dalších komponent výbavy Pro tento krok je potřeba téměř veškerých informací ze struktury vytvořené nástrojem pro stahování konfigurací (znázorněno diagramy na obrázcích 16, 17 a 18). Přenášet však určitě není nutné informace o stupních výbavy a motorizacích, do kterých konkrétní vůz nepatří.

Posledním uchovávaným typem dat je dokument s popisky, který bude ve webové aplikaci využíván pro standardizování pojmenování a překlad kódů typů barev, převodovek, pohonů, karoserií a dalších dat.

3.3.3 Implementace databáze

Konfigurační data modelů se do databáze ukládají do společné kolekce dokumentů ve stejné struktuře, jakou měly třídy (a potažmo i výsledný JSON formát) popisující standardizovaný formát uchovávání konfiguračních dat modelů v sekci 3.2.1. Do struktury byly přidány pouze 2 další údaje. Prvním z nich je atribut *status*, který pomocí svých hodnot určuje stav konfiguračních dat pro daný model (jen dokumenty s hodnotou *active* jsou dostupné k dalším úkonům

popsaným níže). Druhým z přidaných údajů je atribut *added*, který určuje datum a čas přidání konfiguračních dat do databáze.

Samostatnou kolekci zaujímají také již zmíněné dokumenty s popisky a překlady kódů využívaných v konfiguračních datech.

Pro umožnění přístupu ke specifickému výběru dat dle popisu v sekci 3.3.2 bylo využito funkce agregace dat. Tento způsob výběru dat má určité výhody:

- agregace dat probíhají přímo v databázovém systému,
- zpřístupněna jsou pouze specifikovaná data, což snižuje celkový objem přenášených dat,
- oproti pohledům v MongoDB je možné u agregací specifikovat parametry.

Agregace jsou složeny z jednotlivých fází, které jsou specifikovány pomocí MQL. Příklad kódu sloužícího k výběru modelů konkrétní automobilky (ve výpisu je jí Volkswagen) pomocí agregace dat se nachází ve výpisu kódu 5. Kódy sloužící k provedení všech agregací se nachází v příloze D.4. [14]

V první fázi se provádí omezení dokumentů, se kterými se bude dále pracovat, podle hodnoty stavu (aktivní) a kódu automobilky. Ve druhé fázi se seskupují dokumenty dle kódu automobilky a vytváří se tak pole základních informací o modelech automobilky. Do výsledku se vkládají kód a název automobilky. V poslední fázi je výsledek zbaven atributu *_id*.

Pro urychlení získávání výsledků agregací byly pro atributy, podle kterých jsou v různých fázích agregací vybírány relevantní dokumenty (v ukázce kódu agregace jsou to atributy *status* a *brandCode*), vytvořeny indexy.

3.4 API zpřístupňující konfigurační data

Pro přenos konfiguračních dat stojí mezi databází a klientem REST API. Ta slouží pro přenos všech druhů dat popsaných v sekci 3.3.2. Využití REST architektury umožňuje jednoduché škálování této webové služby. Škálování je dále kromě bezstavového charakteru služby podpořeno použitím objemově úsporného formátu JSON pro přenos dat. [9]

API je implementováno pomocí Javy a frameworku Spring Boot, který usnadňuje provádění základní konfigurace.

3.4.1 Parametry dotazů

Jednotlivé druhy dat jsou zpřístupněny pomocí parametrů dotazů na API.

- Souhrn automobilek nevyžaduje speciální parametry.
- Souhrn modelů požaduje kód automobilky.
- Souhrn stupňů výbavy a motorizací vyžaduje kód automobilky a modelu.

- Pro přístup k datům konkrétního vozu je nutný kód automobilky, modelu a vozu.
- Dokument s popisky a překlady kódů je identifikován kódem jazyka popisků.

Jednotlivé parametry jsou předávány pomocí parametrů v URL dotazu poslaného na API. Parametry dotazů jsou před vložením do kódu agregací (přeložených do Javy) nejprve očištěny, aby bylo zamezeno případným útokům typu NoSQL injection. Až poté je kód představující agregace s parametry odeslán na databázový systém, který agregaci provede a vrátí její výsledek.

```
[{
    $match: {
        status: "active",
        brandCode: "volkswagen"
    }
}, {
    $group: {
        _id: {
            brandCode: '$brandCode'
        },
        brandCode: {
            $first: '$brandCode'
        },
        brandName: {
            $first: '$brandName'
        },
        models: {
            $push: {
                code: '$code',
                name: '$name',
                minModelPrice: '$minModelPrice',
                bodyType: '$bodyType',
                modelImage: '$modelImage'
            }
        }
    }
}, {
    $project: {
        _id: 0
    }
}]
```

Výpis 5: Agregace dat pro výběr modelu automobilky Volkswagen

3.4.2 Mezipaměť API

Pro zlepšení výkonu a zmenšení počtu nutných dotazů směřovaných na databázi byla do této API zabudována mezipaměť s pomocí frameworku Caffeine. Mezipaměť byla implementována tak, aby umožňovala jednoduchou konfiguraci chování jednotlivých úložišť, kde každé úložiště je určeno pro jeden druh přenášených dat. [9][15]

Jednotlivým úložištím byly nastaveny maximální počty ukládaných záznamů a záznamy jsou z úložišť mazány po uplynutí časového limitu od doby získání dat z databáze.

3.5 Webový server zpřístupňující webové stránky

Vzhledem k výběru typu zpracování závislostí komponent konfigurací pomocí webové aplikace na straně klienta, bude výhodné využít konfigurační data stahovaná na straně klienta i pro generování obsahu webových stránek. Díky tomu bude webový server méně zatížen, což umožňuje lepší škálování systému pro větší počet uživatelů.

Pro obsluhu dotazů na webovém serveru byl vybrán framework Spring Boot pro Javu. Důvodem tohoto výběru je jednoduchý a rychlý způsob implementace, který tento framework umožňuje. Framework běží na serveru Tomcat.

3.6 Webová aplikace umožňující konfiguraci automobilů

Webová aplikace byla vytvořena pomocí Javascriptu a knihovny jQuery. Kvůli velkému počtu uživatelů, kteří k prohlížení webových stránek používají mobilní zařízení, popř. při prohlížení internetu mezi zařízeními přepínají, je důležitý vývoj flexibilního uživatelského rozhraní, které se přizpůsobí jak standardním velikostem displejů počítačů a notebooků, tak i displejům mobilních zařízení. Z tohoto důvodu byl využit framework Bootstrap verze 4.4.1. [16]

Jednotlivé CSS a JS soubory webových stránek by měly být kvůli snížení objemu přenášených dat k uživateli minifikovány. Kvůli minimalizaci počtu HTTP dotazů by také tyto soubory měly být spojeny do souboru s kaskádovými styly určenými pro danou stránku a druhého souboru zaručujícího její funkčnost. [9]

3.6.1 Návrh jednotlivých kroků cesty uživatele konfigurátorem

Konfigurace se musí skládat z jasně definovaných kroků, aby byly pro uživatele cíle těchto jednotlivých kroků jednoznačné a lehce pochopitelné. Návrh jednotlivých kroků a uživatelského rozhraní byl inspirován analyzovanými konfigurátory. Toto umožňuje uživatelům, kteří se již s podobnými aplikacemi setkali, rychlé osvojení ovládání a navigace dílčími kroky konfigurátoru.

Požadavky na data užívaná v jednotlivých sekcích byly již zmíněny v sekci 3.3.2. Detailní popis některých návrhových detailů jednotlivých kroků je dostupný níže.

3.6.1.1 Výběr automobilky

Tento první krok se samozřejmě v konfigurátorech analyzovaných automobilek nenachází. Při výběru automobilek napomůže k lepší orientaci uživatelů jednoduchý seznam dostupných automobilek, který obsahuje jejich snadno rozpoznatelná loga.

Stejně tak, jak je tomu u většiny automobilek, mělo by i zde existovat rozdělení mezi osobními a užitkovými vozy. Toto rozdělení nabídky znamená pro uživatele jednodušší a přehlednější výběr modelu v dalším kroku konfigurátoru. Uvádění osobních vozů vedle komerčních užitkových vozidel by navíc působilo nepříjemným vizuálním dojmem.

U jednotlivých možností výběru by také měl být udán počet vozů dostupných u dané automobilky, což umožní uživateli jednoduchou volbu mezi většími automobilkami s velkým počtem dostupných modelů a menšími automobilkami s užším výběrem modelů.

3.6.1.2 Výběr modelu vozu

Výběr modelu by měl obsahovat seznam modelů doplněný o jejich grafickou podobu, což usnadňuje uživatelům jejich výběr.

Další nutnou položkou je také možnost omezení výběru, což bude jednoznačnou výhodou zejména u automobilek s velkým výběrem modelů. V tomto kroku konfigurace je asi nejužitečnějším kritériem omezení výběru modelů typ karoserie modelu (např. kombi, SUV, sedan atd.). Pro jednodušší vyhledávání by mělo také být dostupné seřazení modelů dle názvu, pro jednoduché srovnání cen modelů zase řazení modelů dle ceny.

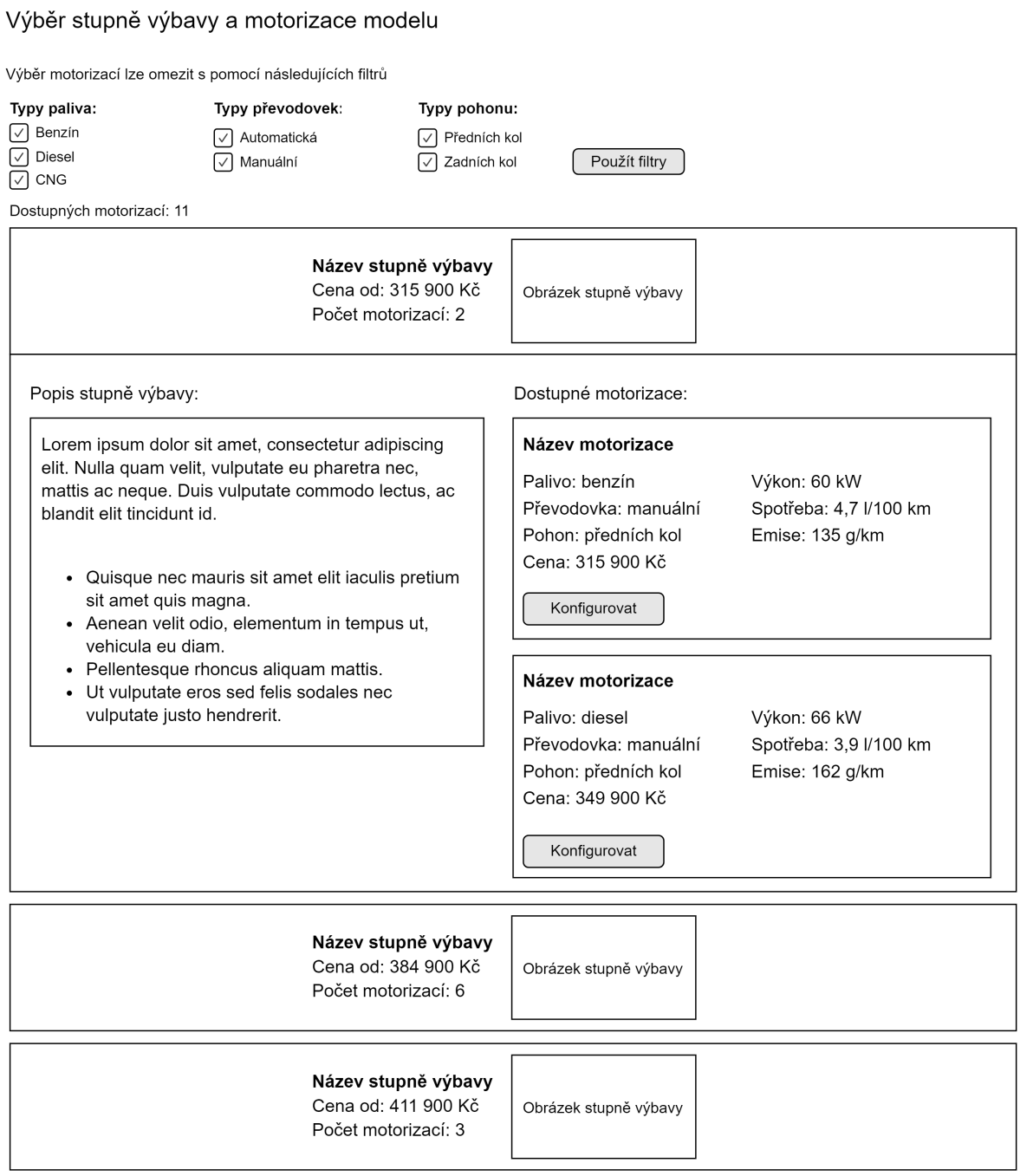
3.6.1.3 Výběr stupně výbavy a motorizace

Ačkoli bývají kroky výběru motorizace a stupně výbavy v konfigurátorech automobilek většinou odděleny a pořadí jejich výběru bývá neměnné, domnívám se, že vizualizace toho, jak výběr motorizace ovlivňuje dostupné možnosti stupňů výbavy (a naopak), nabídne uživatelům větší komfort výběru bez nutnosti přepínání mezi zmíněnými kroky.

Jednotlivé stupně výbavy by měly být seřazeny podle ceny od nejnižší po nejvyšší, což uživateli zřetelně indikuje vybavenost vozu přirozenou formou. Ani v tomto kroku by neměly chybět možnosti omezení výběru - konkrétně dle typu paliva, převodovky a pohonu kol. Popisky jednotlivých motorizací by měly obsahovat jejich základní technické údaje.

Ukázka návrhu grafického rozhraní na obrázku 21 napovídá, že součástí návrhu je využití rozbalovacích prvků pro zobrazení detailního obsahu stupňů výbavy a jejich dostupných motorizací. Tyto prvky umožňují ve sbaleném stavu šetřit místem obrazovky a uživatelé si jejich rozbalením mohou „poznamenat“ možnosti, mezi kterými si vybírají. Výhod tohoto principu zobrazování obsahu je dále využito i při výběru výbavy v dalším kroku konfigurace.

Zobrazování celkového počtu dostupných motorizací a počtu dostupných motorizací pro jednotlivé stupně výbavy umožňuje uživateli na první pohled zjistit, zdali je vozidlo v dané kombinaci zvolených vlastností dostupné, popř. jestli je dostupné ve vybraném stupni výbavy.



Obrázek 21: Návrh rozložení prvků stránky s výběrem stupňů výbavy a motorizací

3.6.1.4 Výběr barvy, kol, interiéru a dalších komponent výbavy

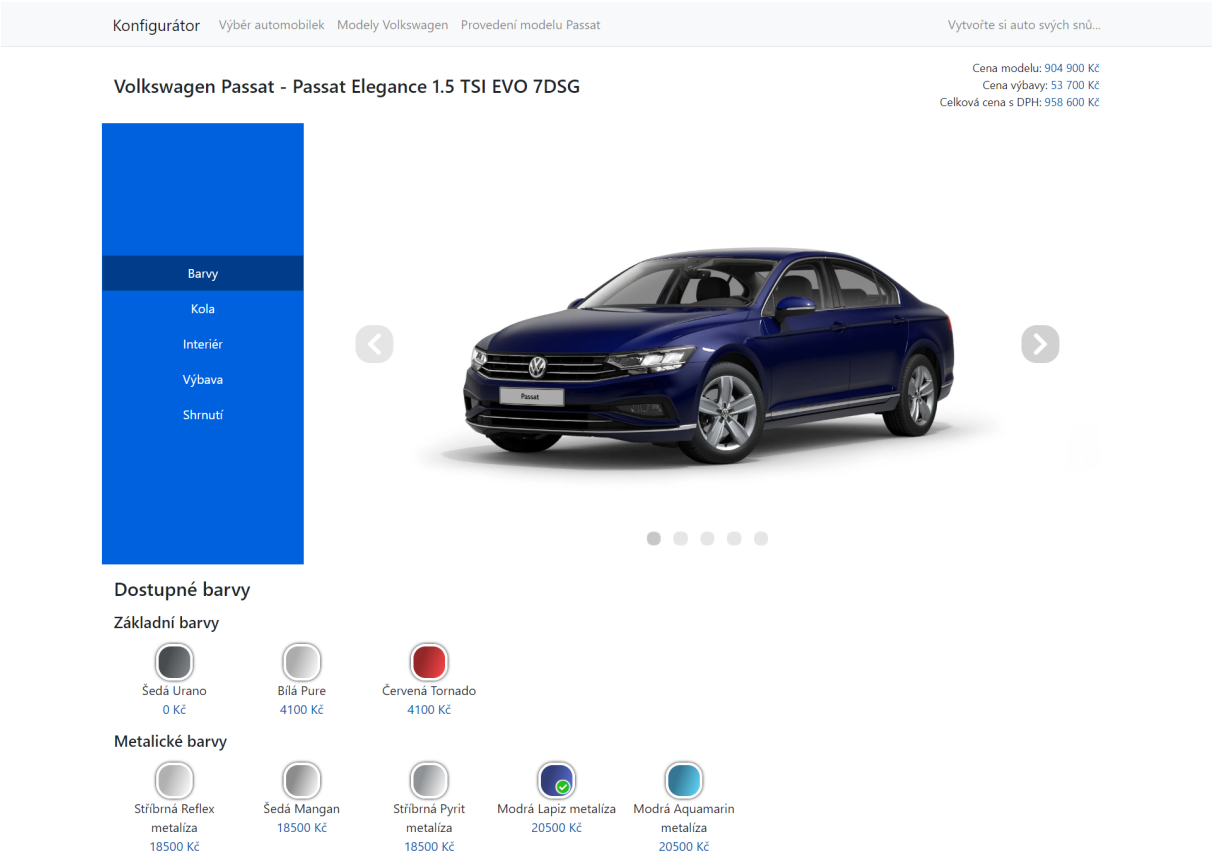
Posledním krokem konfigurace je samotný výběr dílčích komponent vozidla. V analýze konfiguračních aplikací uvedené v knize Knowledge-Based Configuration: From Research to Business Cases [5] byly autory odvozeny základní charakteristiky a poznatky týkající se konfiguračních aplikací.

1. „Navigační panel by měl být umístěn v dominantní pozici na stránce."
2. „Velikost konfiguračních obrázků by měla být dostatečně velká, aby bylo možno vidět detaily."
3. „Možnosti by měly být logicky rozděleny."
4. „Ceny by měly být k vidění ve všech krocích konfigurace."
5. „Aplikace by měly podporovat navigaci dopředu i zpět."
6. „Konfigurátory by měly mít možnost přizpůsobit se při konfiguraci preferencím uživatele."

Návrh konfigurační aplikace tyto body splňuje následujícími způsoby.

1. Navigační panel je lehce viditelný na počátku stránky.
2. Galerie konfiguračních obrázků zaujímá značnou část prostoru stránky, obrázky jednotlivých možností jsou přiměřeně velké.
3. Samotná konfigurace konkrétního vozu je rozdělena do sekcí, možnosti jednotlivých sekcí jsou rozděleny do kategorií (způsob rozdělení komponent do kategorií je v sekci 3.2.1.2).
4. Výsledná cena konfigurace je vždy umístěna v horní části stránky, ceny jednotlivých možností jsou vždy uvedeny v jejich těsné blízkosti.
5. Konfigurátor podporuje navigaci dopředu i zpět.
6. Uživatel může některé kroky přeskočit a poté se k nim vrátit, konfiguraci může začít například i výběrem volitelné výbavy.

Velká pozornost byla věnována hlavním prvkům stránky, které jsou specifické k tomuto kroku konfigurace a zaslouží si zvláštní prioritu umístění na stránce. Jejich výsledný vzhled je na obrázku 22. Navigační panel provádí uživatele konfigurací a graficky jim znázorňuje jejich vzdálenost od cíle - kompletní konfigurace. Stejná část obrazovky na začátku stránky je vždy věnována informacím o ceně aktuální konfigurace. Galerii obrázků zobrazujících výslednou podobu vozu byl dopřán velký prostor na začátku stránky. Galerie není na rozdíl od některých analyzovaných konfigurátorů vždy viditelná na obrazovce uživatele. Důvodem je, že při větší nabídce konfiguračních možností takové galerie ubírají užitečný prostor na obrazovce uživatele při výběru výbavy vozidla.

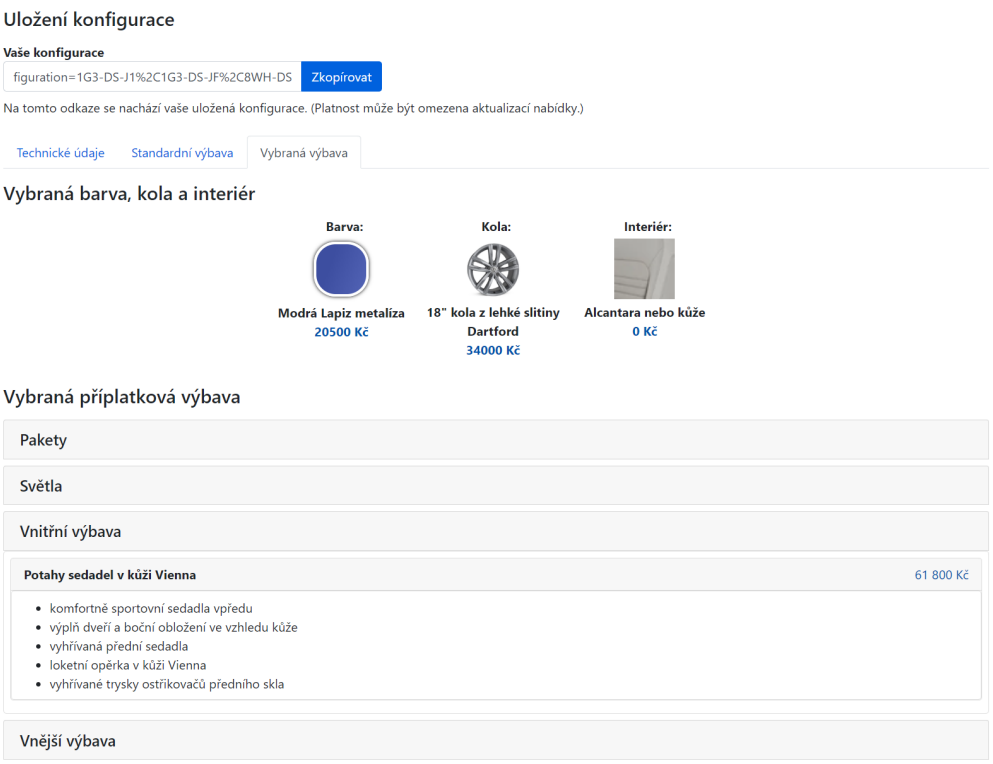


Obrázek 22: Výsledný vzhled hlavních částí konfigurační aplikace

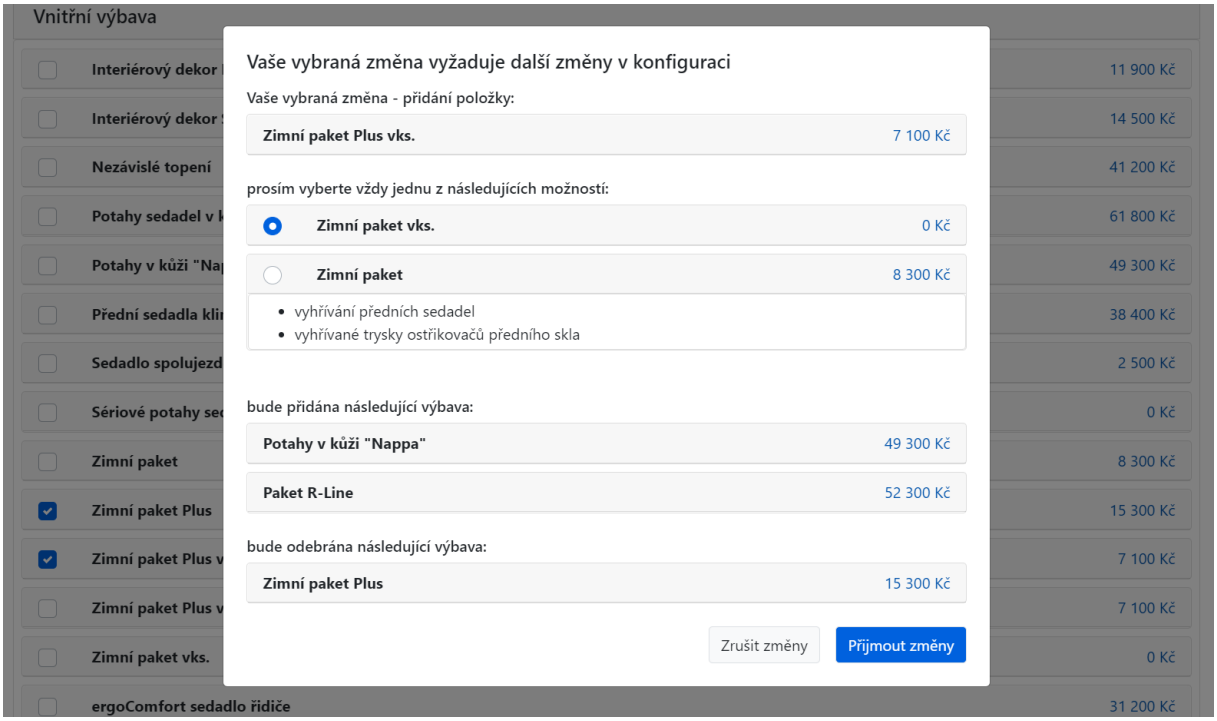
Za zmínku stojí také horní navigační panel zobrazující se i u ostatních kroků konfigurace (s odkazy relevantními ke konkrétnímu kroku konfigurace). Odkazy umístěné na tomto panelu uživatelům umožní jednoduchý přechod k některému z předchozích kroků jejich konkrétní konfigurace.

Shrnutí konfigurace (na obrázku 23) obsahuje sekci s výčtem technických údajů vozu, seznamem standardní výbavy a také samozřejmě sekci se seznamem vybrané volitelné výbavy společně s barvou, koly a interiérem vozu. V této sekci je rovněž explicitně uvedena možnost uložení konfigurace uživatelem.

Důležitou část konfigurací tvoří i dialogy informující uživatele o změnách aktuální konfigurace a umožňující uživatelům volbu požadovaných komponent. Příklad takového dialogu lze vidět na obrázku 24. Na začátku dialogu je uvedena změna, kterou se snaží uživatel provést, poté následuje výběr mezi alternativními možnostmi a výčet změn je ukončen výpisem přidávané a odebírané výbavy. Jedním z cílů grafického návrhu je konzistence vzhledu jednotlivých možností barev, kol, interiérů a volitelné výbavy napříč různými kontexty jejich zobrazení. Jednotlivé položky výbavy jsou proto i zde doplněny rozbalovacím obsahem, který uchovává detailní informace o dané položce výbavy.



Obrázek 23: Shrnutí konfigurace



Obrázek 24: Konfigurační dialog

3.6.2 Implementace funkcionality webové aplikace

Funkcionalita prvních 3 kroků (výběr automobilky, výběr modelu a výběr stupně výbavy s motorizací) je velmi přímočará. Z tohoto důvodu je tato sekce spíše zaměřena na poslední krok konfigurace.

3.6.2.1 Obecný popis funkcionality posledního kroku konfigurátoru

Obecně popsaná funkcionalita konfigurátoru se nachází v aktivitním diagramu na obrázku 25. V dalších sekcích bude věnována pozornost zejména části diagramu zabývající se samotným přidáváním a odebíráním komponent do a z konkrétní konfigurace.

Na začátku se konfigurátor inicializuje buď standardním způsobem nebo načtením uložené konfigurace. Tato funkce aplikace je popsána v sekci 3.6.3. Pokud není k dispozici uložená konfigurace, vyberou se z konfiguračních dat výchozí komponenty - barva, kola a interiér. Výchozí komponenty se vybírají následovně: z možností, u kterých neexistují závislosti na další komponenty se vybere ta, která má nejnižší cenu. Výběr výchozích komponent je prováděn proto, aby byly vizuálně spojeny vybrané komponenty a podoba vozidla v galerii obrázků.

Protože nemusí vždy existovat v konfiguračních datech vozu vhodné výchozí komponenty, které splňují výše uvedenou podmínku, zobrazí se alespoň v tomto případě v galerii obrázků pohledy na exteriér vozu s barvou či koly (popř. oběma komponentami), které se v aktuální konfiguraci nenachází.

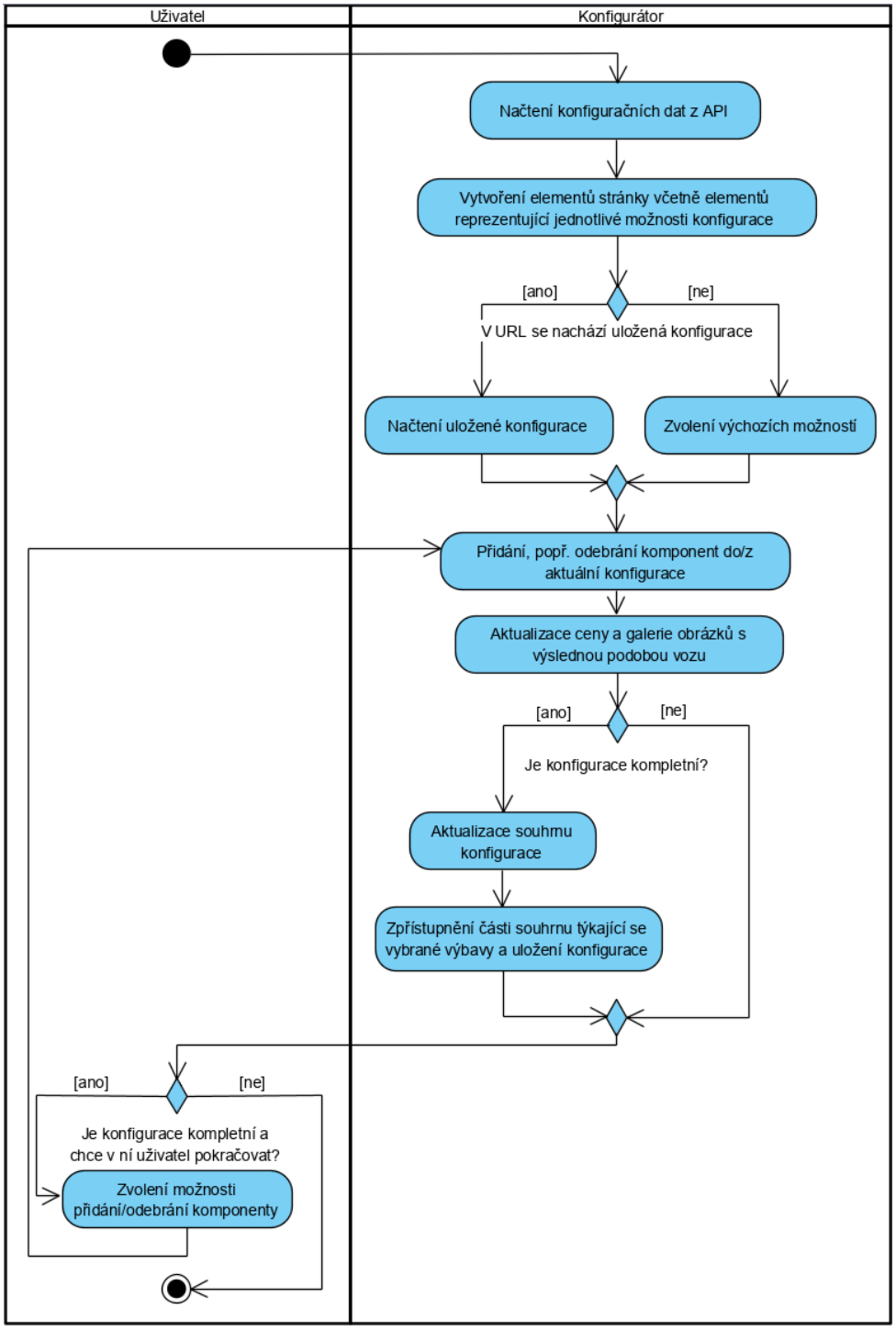
Po přidání či odebrání komponent (tato funkcionalita je popsána v sekcích 3.6.2.4-3.6.2.5) je aktualizována zobrazovaná cena konfigurace a výbavy. Pokud ovlivní změna konfigurace vzhled vozidla, jsou aktualizovány i patřičné obrázky v galerii. Pokud je konfigurace kompletní, aktualizuje se také seznam vybrané výbavy v poslední sekci konfigurátoru. Konfigurace je považována za kompletní, jestliže jsou vybrány barva, kola a interiér.

Výpis shrnutí technických údajů vozu byl implementován s ohledem na možné chybějící technické údaje v konfiguračních datech vozidla, které buďto nebyly v datech automobilek k dispozici a nebyly manuálně doplněny, nebo které se pro dané vozidlo nehodí.

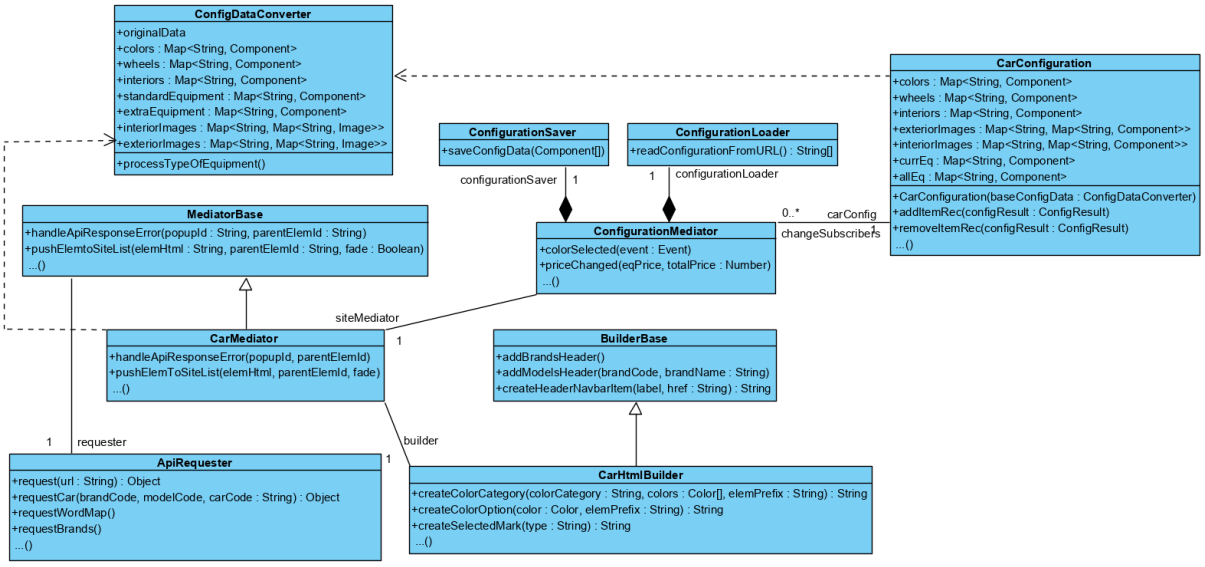
3.6.2.2 Struktura tříd využívaných v implementaci

Stránky reprezentující jednotlivé kroky konfigurace využívají ve své implementaci v jazyku Javascript návrhový vzor prostředník. Tento návrhový vzor pomáhá zmenšit počet vazeb mezi třídami, za jejichž komunikaci je zodpovědný, a také jim pomáhá udržet princip jedné odpovědnosti. U jednotlivých kroků spojuje implementace tohoto návrhového vzoru třídu zodpovědnou za obstarání dat z API a třídu zodpovědnou za tvoření HTML elementů. [17]

Struktura tříd posledního kroku konfigurace (výběr komponent pro konkrétní vůz) je mírně složitější v porovnání s ostatními kroky. Hlavní třídy, které jsou do posledního kroku konfigurace zapojeny, jsou znázorněny v diagramu 26.



Obrázek 25: Diagram aktivit popisující obecný postup konfiguračního procesu



Obrázek 26: Struktura hlavních tříd konfigurační aplikace

Prostředníkem mezi třídou, která získává data z API a třídou, která se stará o tvorbu HTML elementů je v tomto případě *CarMediator*. Mezi třídu *CarConfiguration* starající se o uchovávání konfigurace a o přidávání či odebírání komponent a již zmíněného prostředníka byla vložena další třída (*ConfigurationMediator*), která zprostředkovává komunikaci mezi těmito třídami. Tato třída se také chová jako pozorovatel - třída je upozorňována na změny v konfiguraci a tyto informace pak využívá k volání metod objektu třídy *CarMediator*, který změny zobrazí uživateli.

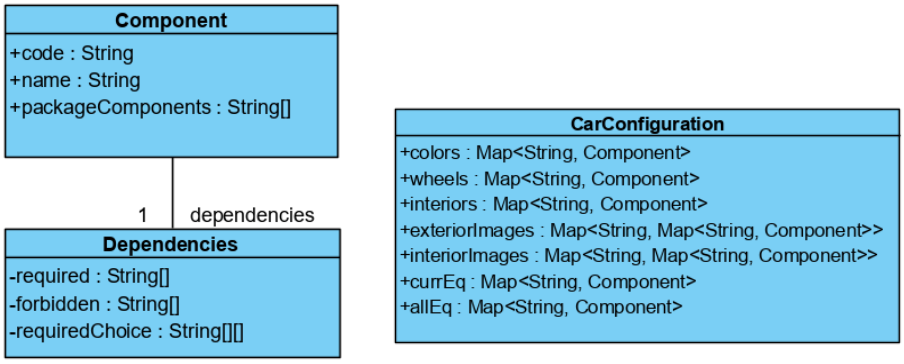
Změny v konfiguraci také ovlivňují třídu *ConfigurationSaver*, která ukládá změny konfigurace do URL. O načítání konfigurací se stará třída *ConfigurationLoader* (více v sekci 3.6.3). Pro základní inicializaci třídy *CarConfigurator* je potřeba její naplnění konfiguračními daty. K tomuto účelu slouží třída *ConfigDataConverter*, která je zodpovědná za převod dat komponent složených z několika částí (z obecných informací a informací specifických pro konkrétní vůz) do jejich sjednocené formy (v diagramu označené jako třída *Component*).

3.6.2.3 Detailní popis dat využívaných při provádění změn konfigurace

Konfigurátor je připravený na nezávislé pořadí výběru různých typů komponent také díky abstrakci komponent, jejichž společné atributy popisuje diagram 27a. Význam atributů této abstrakce byl již popsán v sekci 3.2.1.2 Ukládání konkrétních vozů.

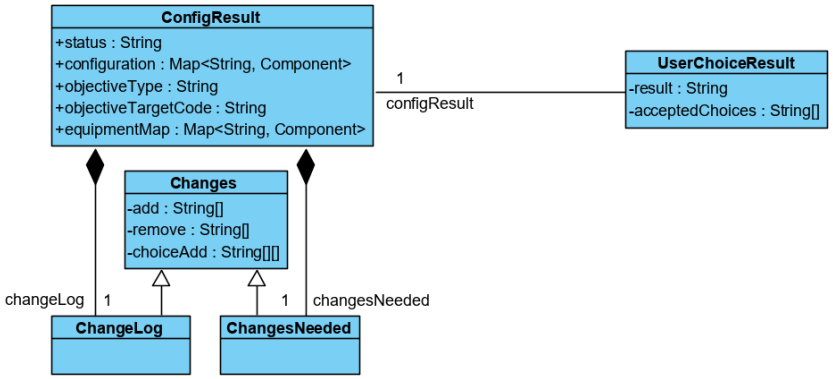
Pro ukládání aktuální konfigurace slouží ve třídě *CarConfiguration* (obr. 27b) objekt označený *currEq*, který je využíván podobně jako mapové struktury běžné u ostatních programovacích jazyků. Díky tomu lze velmi rychle s pomocí kódu komponenty zjistit, zdali je daná komponenta součástí aktuální konfigurace.

Dalšími důležitými strukturami využívanými v konfiguračním procesu jsou objekty tříd *ConfigResult* a *UserChoiceResult*, které jsou k vidění v diagramu 28.



(a) abstrakce komponenty (b) třída uchovávající aktuální konfiguraci

Obrázek 27: Datové třídy využívané v konfiguračním procesu



Obrázek 28: Třídní struktury využívané při provádění změn konkrétní konfigurace

Instance třídy *ConfigResult* v sobě uchovávají kopii aktuální konfigurace. Tato kopie konfigurace je využívána pro testování závislostí při přidávání či odebírání komponent. U komponent, jejichž přidání či odebírání vyžaduje více kroků, si tato kopie konfigurace uchovává stav dosažený doposud provedenými kroky.

Všechny provedené kroky se ukládají také do struktury označené *changeLog*, kam se dle typu provedené akce zapisují kódy přidaných či odebraných komponent. Struktura *changesNeeded* naopak vyznačuje, jaké akce musí být ještě podniknuty pro kompletní vyřešení všech konfliktů závislostí.

Atribut *objectiveType* označuje, zdali původním cílem změny konfigurace bylo přidání či odebrání komponenty, atribut *objectiveTargetCode* kód této původní komponenty udržuje. Objekt *equipmentMap* opět slouží jako mapa komponent, tentokrát z důvodu možného zobrazení informací o komponentách prostřednictvím konfiguračního dialogu uživateli.

Posledním atributem třídy *ConfigResult* je *status*. Ten indikuje pomocí kódu dosavadní stav konfiguračního procesu. Konfigurační proces se může nacházet v 1 ze 3 možných stavů.

1. **Běžný stav** - indikuje normální běh konfigurace. Na konci procesu přidávání či odebírání

komponent tento stav označuje, že není potřeba provádět další úpravy konfigurace. Jedinou potřebnou změnou konfigurace je tedy původní přidání/odebrání komponenty zamýšlené uživatelem.

2. **Vyžadován vstup uživatele** - tento stav označuje, že provedením uživatelem požadované úpravy konfigurace je kromě přidání/odebrání původní komponenty nutné také přidání či odebrání dalších komponent (tento stav samozřejmě vyvolá i nutný výběr jedné z možných alternativ požadovaných komponent).

3. **Neřešitelný stav** - označuje stav, ve kterém není možno pokračovat v řešení konfliktů závislostí konfigurace. Tento stav je detekován, pokud pro daný objekt třídy *ConfigResult* platí alespoň jedna z níže uvedených podmínek.

   (a) V poli *changesNeeded.add* se nachází alespoň jeden z kódů komponent z *changeLog.remove*.

   (b) V poli *changesNeeded.remove* se nachází alespoň jeden z kódů komponent z *changeLog.add*.

   (c) Celý obsah alespoň jednoho z polí v *changesNeeded.choiceAdd* se nachází v *changeLog.remove*.

3.6.2.4 Detailní popis procesů přidávání či odebírání komponent

Proces přidávání či odebírání komponent je popsán sekvenčním diagramem na obrázku 29. Sekvence je vyvolána uživatelem, který provede výběr komponenty, kterou chce do konfigurace přidat nebo z konfigurace odebrat. To vede k inicializaci objektu třídy *ConfigResult*, která udržuje informace o změnách konfigurace. Metoda *handleChanges* zahájí řešení závislostí pomocí rekurzivního algoritmu popsaného v sekci 3.6.2.5. Výsledkem funkce je objekt třídy *ConfigResult* s požadovanými změnami.

Přidávání a odstraňování balíčků komponent (paketů) je řešeno jednoduchým způsobem - při inicializaci objektu *ConfigResult* jsou do polí s požadovanými změnami přidány kódy představující součásti paketů.

Dalším krokem je vyřešení těchto požadovaných změn. Pokud se výsledek řešení závislostí nachází ve stavu, při kterém je nutný vstup uživatele, vygeneruje se konfigurační dialog s popsanými změnami konfigurace. Uživatelova volba je ve formě objektu *UserChoiceResult* předána pomocí zpětného volání do metody *userChoiceResultCallback*. Pokud k provedení změn není potřeba vstupu uživatele, je zmíněná metoda volána přímo, stav objektu *UserChoiceResult* je v tomto případě vždy nastaven na přijatý. Pokud se nepodařilo doporučit řešení změn aktuální konfigurace, které by vedly k validní konfiguraci, je vyvolána výjimka a původní změny konfigurace zamýšlené uživatelem nejsou provedeny.

V dalším kroku procesu se zachází s objektem třídy *UserChoiceResult*, který nyní obsahuje doporučené změny konfigurace a uživatelovy volby. Pokud uživatel změny zamítnul v konfigurač-

ním dialogu, změny nejsou provedeny. Pokud jsou změny akceptovány, záleží další vývoj na tom, zdali byla součástí konfiguračního dialogu volba mezi alternativními komponentami.

Pokud výběr z alternativních komponent nebyl součástí dialogu, jsou postupně všechny změny převedeny do aktuální konfigurace. Pokud ano, je pro každou uživatelem zvolenou komponentu spuštěna metoda *addOrRemoveItem* (v diagramu označená 1.1). Tato metoda vrací pravdivostní hodnotu indikující, zdali bylo provedeno požadované přidání/odebrání komponenty. Jestliže je úspěšně provedeno přidání všech komponent uživatelem vybraných z dostupných alternativ, přidá se do aktuální konfigurace také komponenta, která tento výběr z alternativních komponent vyvolala. V opačném případě metoda *userChoiceResultCallback* končí a sama vrací hodnotu *false*, která je zpět propagována do metody *addOrRemoveItem*.

Způsobené změny konfigurace vyvolají ve třídě *CarConfiguration* volání na instanci *ConfigurationMediator*, která se postará o to, aby instanci třídy *CarMediator* byla předána veškerá data potřebná k zobrazení změn konfigurace uživateli (na diagramu je toto zobecněné volání označeno *1.5 updateSelectedItems*).

3.6.2.5 Detailní popis algoritmu generujícího řešení konfigurace v závislosti na požadovaných změnách

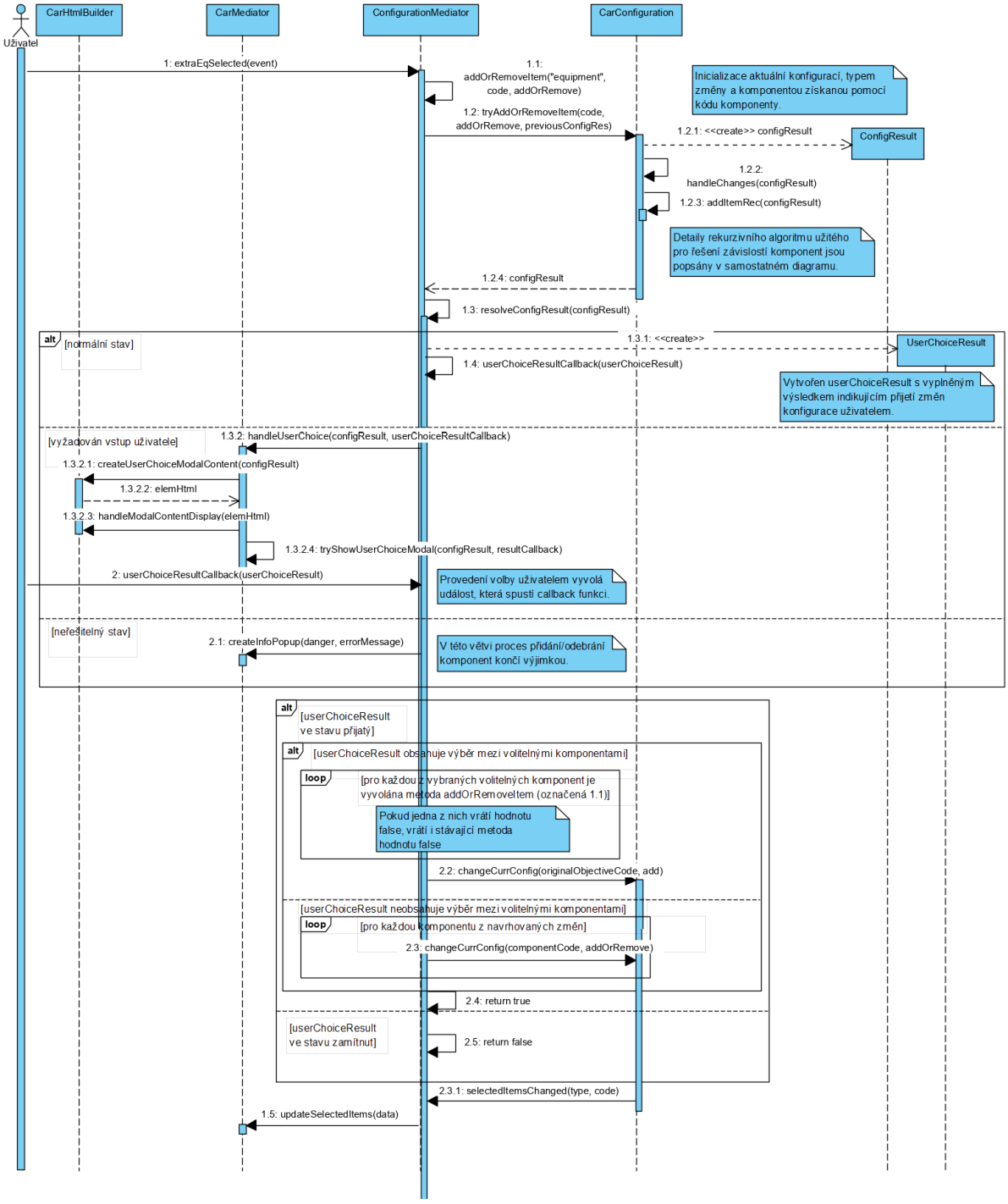
Při přidávání nových komponent do konkrétní konfigurace je nutné kontrolovat všechny 3 typy závislostí uvedené v sekci 3.2.1.2. Přidávání komponent může vyvolat přidávání dalších komponent, výběr z možných alternativ, či odebrání nekompatibilních komponent.

Při odebírání komponent je nutné kontrolovat pouze 2 typy závislostí mezi komponentami - požadavky na vložení dalších komponent a výběr mezi požadovanými komponentami. Kontrolovat nekompatibilitu mezi komponentami je v tomto bodě již neúčelné, jelikož je komponenta odebírána a předpokládá se, že vytvořená konkrétní konfigurace je konzistentní (podmínky určené závislostmi všech komponent jsou splněny).

Pokud určitá komponenta v konfiguraci vyžaduje přítomnost odebírané komponenty v konfiguraci, musí být odebrána. Pokud odebíraná komponenta zaručuje jedné či více komponentám v aktuální konfiguraci svou přítomností splnění podmínky výběru alespoň jedné z alternativních komponent, existují 2 možnosti řešení. První možností je nabídnout uživateli přidání jedné ze zbylých alternativních komponent. Druhou možností je odebrání komponenty, která závisí na odebírané komponentě. Toto řešení (používané např. konfigurátorem značky Volkswagen), nabízí určitou výhodu při implementaci konfiguračního systému, a to: odebírání komponent může na rozdíl od přidávání komponent vyvolat pouze další odebírání komponent.

Počátek algoritmu záleží na tom, zdali je původním cílem změny konfigurace přidávání či odebírání komponent. Kód funkce *handleChanges* sloužící jako počátek algoritmu je uveden v ukázce kódu 6.

Pokud je nutné přidání jedné či více komponent, spustí se rekurzivní funkce addItemRec (ukázka kódu 7).



Obrázek 29: Sekvenční diagram procesu přidání/odebrání komponenty uživatelem

Pro řešení změn konfigurace, u kterých je nutné provádět odstranění komponent, slouží rekurzivní funkce removeItemRec (ukázka kódu 8).

```
handleChanges(configResult)
{
    //Počátek závisí na povaze první původní vyvolané změny konfigurace
    if(configResult.objectiveType == "add")
        configResult = this.addItemRec(configResult);
    else
        configResult = this.removeItemRec(configResult);

    if(configResult.status == configResult.STATUS_UNABLE)
        return configResult;

    //Nastavení stavu, kdy je vyžadován vstup uživatele
    if(this.isUserInputNedeed(configResult))
        configResult.status = configResult.STATUS_USER_INPUT_NEEDED;

    return configResult;
}
```

Výpis 6: Vstupní bod algoritmu generujícího řešení změn konfigurace

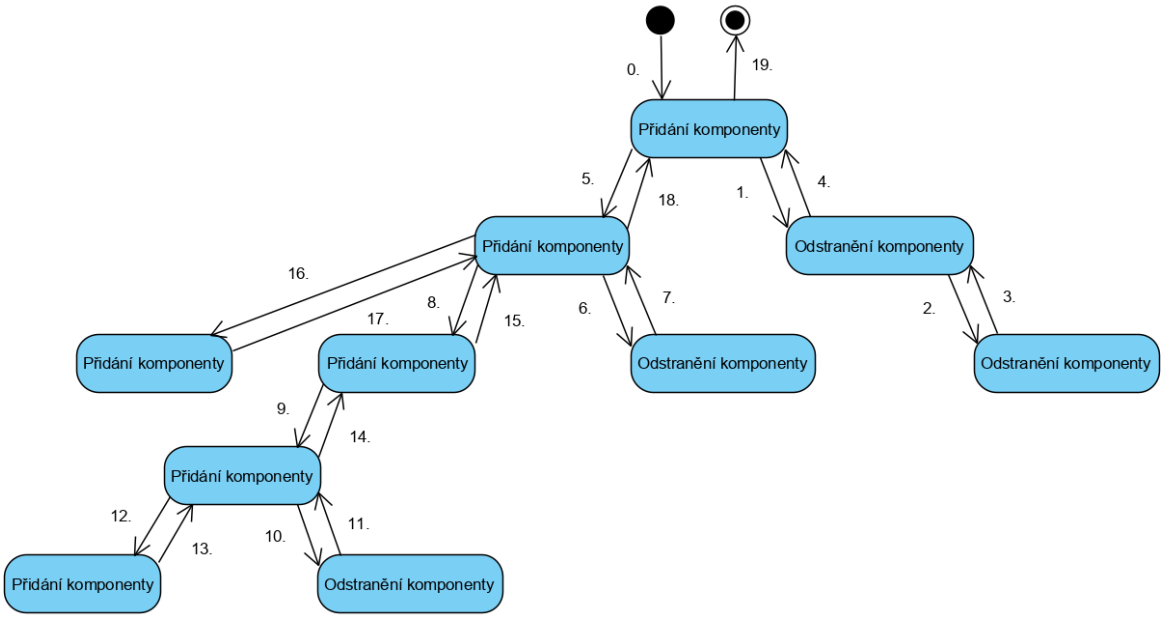
V diagramu 30 lze vidět stromovou strukturu a pořadí volání a návratů rekurzivních funkcí při vkládání komponenty, která svými závislostmi spouští řetězovou reakci přidávání a odebírání dalších komponent.

3.6.3 Ukládání konkrétních konfigurací

Pro ukládání konkrétních konfigurací byla vybrána metoda využívající aktualizaci URL parametrů při změně konfigurace. Tento způsob ukládání je hojně využívaný a v mnoha ohledech pohodlný jak pro uživatele, tak i z hlediska implementace, která není závislá na externích systémech (porovnání způsobů ukládání konfigurací se nachází v sekci 2.2.10).

Konfigurační systém vždy při změnách konfigurace aktualizuje dotazové parametry v URL webové stránky, což se okamžitě projeví v adresním řádku prohlížeče uživatele. Uživateli je v poslední sekci konfigurace explicitně nabídnuto zkopírovaní odkazu konkrétní konfigurace, avšak tento odkaz se neodlišuje od toho zobrazovaného v adresním řádku prohlížeče.

Konkrétní konfigurace je v parametru URL reprezentována jako abecedně seřazený seznam kódů komponent konkrétní konfigurace, kde je každý kód oddělený pomocí speciálního znaku. Při inicializaci konfiguračního systému jsou komponenty reprezentované kódy v URL opět přidány do aktuální konfigurace.



Obrázek 30: Pořadí volání a návratů rekurzivních funkcí při vkládání komponenty

3.6.4 Testování korektnosti konfigurací

Konfigurační systém byl manuálně testován pomocí porovnání výsledků (i dílčích) konfiguračního procesu a výsledných konfigurací s oficiálními konfigurátory automobilek. Testování bylo zaměřeno hlavně na přidávání a odebírání komponent s mnoha závislostmi, které spouštěly řetězové reakce přidávání a odebírání dalších komponent.

Kromě rozložení informací do jednotlivých konfiguračních dialogů (což negativně neovlivňuje informovanost uživatele), byly výsledky zpracování závislostí stejné [2].

[2]Stejně se konfigurační systém choval také při zpracování patrně chybně nastavených závislostí komponent automobilky - po přidání komponenty s názvem 18″ kola z lehké slitiny Bonneville při konfiguraci vozu s označením Passat Business 1.5 TSI EVO 6G je uživateli nabídnuta volba mezi rezervním dojezdovým kolem a plnohodnotným 17″ rezervním kolem. Výběr 17palcového rezervního kola pak vyžaduje výběr některých ze 17palcových kol, což opět odstraní původně vybraná 18palcová kola.

4 Závěr

Jedním z hlavních přínosů práce je provedená analýza konfiguračních možností 20 automobilek (a jejich 19 webových konfigurátorů). Výsledkem této analýzy byly poznatky týkající se společných rysů těchto konfigurátorů a také jejich rozdílů plynoucích z různých přístupů automobilek a způsobů prezentace svých vozů.

Po provedené analýze konfigurátorů bylo zřejmé, že standardizovaná struktura určená k udržení konfiguračních dat různých automobilek nebude moci existovat bez malých ústupků. Vzniklý návrh této struktury však umožňuje využívat konfigurační data automobilů bez ohledu na značku vozu, což je jeho nesporná výhoda.

Dále byl vytvořen nástroj pro získávání dat přímo od automobilek Audi, Seat a Volkswagen a následné převádění těchto dat do navržené struktury. Součástí převádění dat byla také standardizace názvů kategorií komponent a technických údajů, což umožňuje jednodušší porovnání různých druhů a značek automobilů. Celkem byla s pomocí tohoto nástroje získána a do navržené unifikované struktury převedena data 60 modelů automobilů, z nichž je 47 modelů osobních vozů a 13 modelů vozů užitkových.

Posledním přínosem této práce je využití všech předchozích výsledků k vytvoření konfigurační webové aplikace v mnoha ohledech srovnatelné s konfiguračními aplikacemi automobilů dostupnými na českém trhu. Popis tvorby konfiguračního systému je doprovázen příručkou obsahující doporučení technologií vhodných pro budování konfiguračních systémů.

Literatura

1. *All Configurators of the Configurator Database* [online]. Vienna: CyLEDGE Media, 2018 [cit. 2020-04-17]. Dostupné z: `https://www.configurator-database.com/configurators%5C#/`.
2. CHINNAIAH, Pratap; KAMARTHI, Sagar. Mass Customization and Manufacturing. In: *Innovations in Competitive Manufacturing*. 1. vyd. Boston, Massachusetts: Kluwer Academic Publishers, 2000, s. 283–296. ISBN 0-7923-7896-2.
3. *Mass customization* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-04-17]. Dostupné z: `https://en.wikipedia.org/wiki/Mass_customization`.
4. SABIN, Daniel; WEIGEL, Ranier. Product Configuration Frameworks-A Survey. *IEEE Intelligent Systems and their Applications*. 1998, roč. 1998, č. 4, s. 42–49. Dostupné z DOI: `10.1109/5254.708432`.
5. FELFERNIG, Alexander; BAGLEY, Claire; HOTZ, Lothar; TIIHONEN, Juha. *Knowledge-Based Configuration: From Research to Business Cases*. 1. vyd. Burlington, Massachusetts: Morgan Kaufmann, 2014. ISBN 9780124158177.
6. *Registrace nových osobních automobilů v ČR za rok 2019* [online]. Praha [cit. 2020-02-22]. Dostupné z: `https://www.sda-cia.cz/repository-volnedostupna`.
7. *WebP* [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-04-24]. Dostupné z: `https://en.wikipedia.org/wiki/WebP`.
8. HU, Jinyan; SONG, Shaojing; GONG, Yumei. Comparative Performance Analysis of Web Image Compression. In: *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)* [online]. Shanghai, 2017, s. 1–5 [cit. 2020-04-24]. Dostupné z: `https://ieeexplore.ieee.org/abstract/document/8301939`.
9. SHIVAKUMAR, Shailesh Kumar. *Architecting High Performing, Scalable and Available Enterprise Web Applications*. 1st Edition. Burlington, Massachusetts: Morgan Kaufmann Publishers, 2014. ISBN 9780128022580.
10. NURSEITOV, Nurzhan; PAULSON, Michael; REYNOLDS, Randall; IZURIETA, Clemente. Comparison of JSON and XML data interchange formats: A case study. In: *22nd International Conference on Computer Applications in Industry and Engineering 2009 (CAINE-2009)* [online]. 2009, s. 157–162 [cit. 2020-04-24]. Dostupné z: `https://www.semanticscholar.org/paper/Comparison-of-JSON-and-XML-Data-Interchange-A-Case-Nurseitov-Paulson/84321e662b24363e032d680901627aa1bfd6088f`.

11. ZUNKE, Saurabh; D'SOUZA, Veronica. JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats. In: *International Journal of Computer Science and Network* [online]. Volume 3, Issue 4. 2014, s. 257–261 [cit. 2020-04-24]. Dostupné z: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.644.7894%5C&rep=rep1%5C&type=pdf`.

12. RAUT, A. B. NOSQL Database and Its Comparison with RDBMS. In: *International Journal of Computational Intelligence Research* [online]. Volume 13, Number 7. Delhi: Research India Publications, 2017, s. 1645–1651 [cit. 2020-05-03]. Dostupné z: `http://www.ripublication.com/ijcir17/ijcirv13n7_08.pdf`.

13. PLECHAWSKA-WOJCIK, Malgorzata; RYKOWSKI, Damian. Comparison of Relational, Document and Graph Databases in the Context of the Web Application Development. In: *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology*. 1st ed. New York: Springer, 2016, s. 3–13. ISBN 978-3-319-28559-7. Dostupné z DOI: `10.1007/978-3-319-28561-0_1`.

14. *The MongoDB 4.2 Manual* [online] [cit. 2020-05-02]. Dostupné z: `https://docs.mongodb.com/manual/`.

15. DHALL, Chander. *Scalability Patterns: Best Practices for Designing High Volume Websites*. 1st ed. New York: Apress, 2018. ISBN 9781484210741.

16. GASSTON, Peter. *The Modern Web: Multi-Device Web Development with HTML5, CSS3, and JavaScript*. 1st ed. San Francisco: No Starch Press, 2013. ISBN 9781593274870.

17. *The Catalog of Design Patterns* [online]. c2014-2020 [cit. 2020-04-20]. Dostupné z: `https://refactoring.guru/design-patterns/catalog`.

18. DEARI, Raif; ZENUNI, Xhemal; AJDARI, Jaumin; ISMAILI, Florije; RAUFI, Bujar. Analysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL. In: *2018 International Conference on Information Technologies (InfoTech)* [online]. IEEE, 2018, s. 1–4 [cit. 2020-05-03]. ISBN 978-1-5386-9520-3. Dostupné z DOI: `10.1109/InfoTech.2018.8510719`.

19. PETKOVIC, Dusan. JSON Integration in Relational Database Systems. In: *International Journal of Computer Applications* [online]. 168. vyd. 2017-06-15, sv. 168, s. 14–19 [cit. 2020-05-03]. Č. 5. ISSN 09758887. Dostupné z DOI: `10.5120/ijca2017914389`.

20. TANG, Enqing; FAN, Yushun. Performance Comparison between Five NoSQL Databases. In: *2016 7th International Conference on Cloud Computing and Big Data (CCBD)* [online]. IEEE, 2016, s. 105–109 [cit. 2020-05-03]. ISBN 978-1-5090-3555-7. Dostupné z DOI: `10.1109/CCBD.2016.030`.

21. KUMAR, K. B. Sundhara; SRIVIDYA; MOHANAVALLI, S. A performance comparison of document oriented NoSQL databases. In: *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)* [online]. IEEE, 2017, s. 1–6 [cit. 2020-05-03]. ISBN 978-1-5090-3716-2. Dostupné z DOI: 10.1109/ICCCSP.2017.7944071.

22. *MongoDB Licensing* [online]. c2020 [cit. 2020-05-03]. Dostupné z: https://www.mongodb.com/community/licensing.

23. *Apache CouchDB* [online]. Forest Hill, Maryland: Apache Software Foundation, c2020 [cit. 2020-05-03]. Dostupné z: https://couchdb.apache.org/.

24. *Apache CouchDB® 3.0.0 Documentation* [online]. Forest Hill, Maryland: Apache Software Foundation, c2020 [cit. 2020-05-03]. Dostupné z: https://docs.couchdb.org/en/stable/index.html.

25. LORENC, Paweł; WODA, Marek. IaaS vs. Traditional Hosting for Web Applications - Cost Effectiveness Analysis for a Local Market. In: *Advances in Dependability Engineering of Complex Systems* [online]. Cham: Springer International Publishing, 2018, s. 233–243 [cit. 2020-05-03]. ISBN 978-3-319-59414-9. Dostupné z DOI: 10.1007/978-3-319-59415-6_23.

26. *Round 18 results - TechEmpower framework benchmarks* [online]. 2019 [cit. 2020-05-04]. Dostupné z: https://www.techempower.com/benchmarks/%5C#section=data-r18%5C&hw=ph%5C&test=json.

27. *Project Information Framework Tests Overview - TechEmpower/FrameworkBenchmarks* [online]. c2020 [cit. 2020-05-04]. Dostupné z: https://github.com/TechEmpower/FrameworkBenchmarks/wiki/Project-Information-Framework-Tests-Overview.

28. PEKARSKY, Max. *Does your web app need a front-end framework?* [online]. New York: Stack Overflow, c2020 [cit. 2020-05-04]. Dostupné z: https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/.

29. SAKS, Elar. *JavaScript frameworks: Angular vs React vs Vue*. Helsinki, 2019. Bakalářská práce. Haaga-Helia University of Applied Sciences.

30. WOHLGETHAN, Eric. *Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js*. Hamburg, 2018. Bakalářská práce. Hamburg University of Applied Sciences.

A Zdrojový kód rekurzivní funkce řešící přidávání komponent do aktuální konfigurace

```
addItemRec(configResult)
{
    //Výběr přidávané komponenty, pole changesNeeded používaná jako zásobníky
    let extraEq = this.allEq[configResult.changesNeeded.add[0]];

    //Získá kódy komponent, které jsou součástí aktuální konfigurace,
    //ale s přidávanou komponentou jsou nekompatibilní
    let forbiddenEquipped = this.getForbiddenEquippedItems(configResult.
        configuration, extraEq.dependencies.forbidden);

    //Získá kódy komponent, které jsou přidávanou komponentou vyžadovány,
    //ale nejsou součástí aktuální konfigurace
    let requiredNotEquipped = this.getRequiredNotEquippedItems(configResult.
        configuration, extraEq.dependencies.required);

    //Získá pole polí kódů komponent reprezentující výběry alternativních
    //komponent, které nejsou konkrétní konfigurací splněny
    let requiredChoicesLeft = this.getRequiredChoices(configResult.
        configuration, extraEq.dependencies.requiredChoice);

    //Odebere přidávanou komponentu z pole nutných změn, přidá přidávanou
    //komponentu do prozatimní reprezentace konkrétní konfigurace a upraví
    //záznam provedených změn
    configResult.changesNeeded.add.removeByValue(extraEq.code);
    configResult.configuration[extraEq.code] = extraEq;
    configResult.changeLog.add.pushIfUnique(extraEq.code);

    //Sjednocení již známých nutných změn s nově objevenými potřebnými změnami
    configResult.changesNeeded.add.mergeWith(requiredNotEquipped);
    configResult.changesNeeded.remove.mergeWith(forbiddenEquipped);
    configResult.changesNeeded.choiceAdd.mergeWith(requiredChoicesLeft);

    //Kontrola a případné nastavení neřešitelného stavu uvnitř metody
    if(this.checkIfUnresolvable(configResult))
        return configResult;
```

```
    //Rekurzivní volání metody - odstranění komponent následováno přidáváním
    //komponent
    while(configResult.changesNeeded.remove.length !== 0)
    {
        configResult = this.removeItemRec(configResult);
    }
    while(configResult.changesNeeded.add.length !== 0)
    {
        configResult = this.addItemRec(configResult);
    }

    return configResult;
}
```

Výpis 7: Rekurzivní funkce řešící přidávání komponent do aktuální konfigurace

B Zdrojový kód rekurzivní funkce řešící odebírání komponent z aktuální konfigurace

```
removeItemRec(configResult)
{
    //Výběr odebírané komponenty
    let extraEq = this.allEq[configResult.changesNeeded.remove[0]];
    //Odebrání komponenty lze provést jen pokud je komponenta součástí
    //konkrétní konfigurace a zároveň není součástí standardní výbavy vozidla
    if(this.standardEq[extraEq.code] == null && configResult.configuration[
        extraEq.code] != null)
    {
        //Získání pole kódů komponent, které požadují přítomnost odebírané
        //komponenty v konkrétní konfiguraci, metoda může také nastavit stav
        //výsledku na neřešitelný, pokud by bylo požadováno odebrání komponenty
        //standardní výbavy
        let removedPartOfRequired = this.getRemovedDependencies(configResult,
            extraEq);
        if(configResult.status == configResult.STATUS_UNABLE)
            return configResult;

        //Odebrání odebírané komponenty z prozatimní reprezentace konfigurace,
        //odebrání kódu odebírané komponenty z požadovaných změn, zapsání
        //provedení změny do logu a přidání nově objevených požadovaných
        //odběrů komponent
        delete configuration[extraEq.code];
        changesNeeded.remove.removeByValue(extraEq.code);
        changeLog.remove.pushIfUnique(extraEq.code);
        changesNeeded.remove.mergeWith(removedPartOfRequired);

        if(this.checkIfUnresolvable(configResult))
            return configResult;
```

```
        //Pokračování v rekurzi - odstraňování dalších komponent
        while(configResult.changesNeeded.remove.length !== 0 && configResult.
            status != configResult.STATUS_UNABLE)
        {
            configResult = this.removeItemRec(configResult);
        }

        return configResult;
    }
    else //V tomto případě nelze komponentu odstranit
    {
        configResult.status = configResult.STATUS_UNABLE;
        return configResult;
    }
}
```

Výpis 8: Rekurzivní funkce řešící odebírání komponent z aktuální konfigurace

C Teoretická příručka volby technologií pro budování konfiguračních systémů

C.1 Úvod

Výběr systémů a technologií využívaných konfiguračním systémem může radikálně ovlivnit jeho výkon, škálovatelnost, obtížnost implementace a výši finančních prostředků potřebných k uvedení systému do provozu.

Cílem této příručky je nastínit několik možností výběru technologií, které jsou k budování konfiguračního systému vhodné.

C.2 Databáze

Databáze konfiguračního systému by měla být vybrána s ohledem na tato hlavní kritéria:

- rychlost výběru dat,
- úspornost ukládání dat,
- umožnění škálování databázového systému,
- cenu licence,
- rychlost a jednoduchost implementace.

C.2.1 Výběr databázového systému

Při výběru databáze se okamžitě nabízí výběr jedné z dokumentově orientovaných databází. Dokumentově orientované databáze byly stvořeny pro efektivní ukládání a načítání JSON dokumentů, jako například těch, ve kterých je možné efektivně ukládat konfigurační data. Dokumentově orientované databáze nabízejí vysoký výkon při čtení záznamů (obzvláště při velkém souběžném zatížení) a jejich implementace je ve srovnání s relačními databázemi jednoduchá. Dokumentově orientované databázové systémy jsou navíc v mnohých případech považovány za lépe škálovatelné, udržitelné a méně finančně náročné. [12] [13] [18]

Výběr relačního databázového systému není podpořen jejich obtížnější implementací, přičemž musí být s konfiguračními daty ve formátu JSON buďto manipulováno speciálním způsobem (manipulace s daty ve formátu JSON je u jednotlivých relačních databázových systémů více či méně podporovaná [19]), nebo musí být konfigurační data přemapována do podoby využitelné v relačních databázových systémech. Rozhodujícím kritériem při výběru databázového systému však může také být existující infrastruktura či již zakoupené licence.

C.2.2 Příklady vhodných databázových systémů

C.2.2.1 MongoDB

Velmi výkonný dokumentově orientovaný databázový systém, který se často umisťuje na předních příčkách srovnání rychlosti čtení a zápisu dokumentů mezi dokumentově orientovanými databázemi. [20] [21]

MongoDB také podporuje horizontální škálování systému pomocí distribuce dat mezi jednotlivé servery. [14]

Aktuální verze MongoDB jsou vydávány pod licencí SSPL. Tato licence umožňuje využívání a modifikaci MongoDB zdarma až do doby, kdy je služba využívající MongoDB přístupná třetím stranám. V tu chvíli je nutné zpřístupnění kódu služby veřejnosti nebo zakoupení komerční licence. [22]

C.2.2.2 CouchDB

Ještě volnější licenci v porovnání s MongoDB využívá dokumentově orientovaný databázový systém CouchDB. Je totiž vydáván pod licencí Apache License 2.0, která povoluje využívání a modifikaci CouchDB bez nutnosti platby. [23]

V porovnání výkonu CouchDB mírně zaostává za MongoDB [21]. CouchDB však také podporuje podobný systém horizontálního škálování [24].

C.3 Využití cloudových služeb

Využití cloudových služeb pro potřeby databáze, ale i dalších systémů přináší oproti tradičnímu či vlastnímu hostování systémů následující výhody:

- flexibilní škálování dle aktuální potřeby,
- bezkonkurenční dostupnost a spolehlivost,
- nulovou počáteční investici do infrastruktury,
- jednodušší a rychlejší nasazení systémů.

Cloudové služby využívají model zpoplatnění, kde se platí za využité zdroje za časový interval. To může být stejně jako u tradičního hostingu díky rozložení nákladů velkou výhodou. Oproti tradičnímu hostingu však cloudové služby mohou být kvůli nabízeným výhodám o něco dražší. [9] [25]

C.4 Doručování statického obsahu

Pro lepší škálovatelnost konfiguračního systému a případné zrychlení doručování statického obsahu jako jsou JS a CSS soubory či obrázky je možné využití Content Delivery Network (CDN). Lepší škálovatelnost je dosažena rozložením celkové zátěže - spojené s plněním dotazů ohledně

získání obsahu - na více serverů. Zrychlení doručování obsahu je pak možno docílit dalším cachováním a inteligentním směrováním dotazů na nejbližší dostupný server. [9]

Použití CDN v infrastruktuře konfiguračního systému je volitelným krokem. Ukládání obrázků výsledných podob vozů je však pro nabízení konfigurace některých automobilek pomocí standardizované struktury konfiguračních dat nutné. Jedná se o ty automobilky, které využívají pro zobrazení podoby vozu více vrstev obrázků formátu PNG.

U systému používaného k doručování obrázků je vhodné pro zrychlení přenosu a zmenšení objemu přenášených obrázků také využít funkci, která umožňuje v dotazu na obrázek specifikovat jeho požadovanou velikost. [9] [16]

C.5 Webové frameworky na straně serveru

Porovnání webových frameworků užívaných na straně serveru podle výkonu nemusí být kvůli velkému počtu proměnných, které mohou ovlivňovat výsledky, vždy velmi přesné. Ke srovnání výkonu webových frameworků je možné použít testy provedené stránkou TechEmpower [26]. Z důvodů odstranění vlivu některých nežádoucích proměnných (jako je například získání dat z databáze) a měření výkonu základních funkcí, které bude framework plnit jako součást konfiguračního systému, je pro srovnání vhodné použít výsledky testu nazvaného *JSON serialization*. Tento test se zaměřuje na počet zpracování dotazů za časovou jednotku. Zpracování dotazu se skládá z přečtení požadované akce dotazu, serializace jednoduchého objektu do formátu JSON a jeho následného odeslání. [27]

Na předních příčkách[3] výsledků testů se často umisťovaly frameworky jazyků C++, Rust, Java a Go. Příklady nejvýkonnějších frameworků těchto jazyků jsou následující:

- Ulib (C++)
- Hyper a Actix (Rust)
- Proteus, Firenio (Java)
- Atreugo (Go)

Ze známějších frameworků se poměrně dobře umístily také ASP.NET Core a Node.js. Populární frameworky oblíbené i díky své jednoduché implementaci (např. Spring a Django) se umístily na nižších pozicích. [26]

Výkon samotných frameworků a hardwaru, na kterém běží, by však měl být rovněž podpořen implementací mezipaměti, což může ulehčit zátěž vyvíjenou na databázi [9]. Dalším důležitým kritériem je jakákoli předešlá zkušenost s frameworkem, popř. s programovacím jazykem, pro který je framework postaven. To může mít za výhodu urychlení implementace stejně jako výběr frameworku s dobrou podporou, a to ať už přímou - od vývojářů pomocí detailní

[3] Pozice s výkonem 90-100% v porovnání s vítězem testu.

dokumentace - či nepřímou - od komunity vyvíjející v konkrétním frameworku. Z hlediska podpory vynikají většinou populárnější frameworky, pro které je také vyvíjeno více specializovaných knihoven a nástrojů (například právě pro správu mezipaměti).

Zmíněné webové frameworky nabízejí poměrně volné licence, mezi které se řadí např. licence MIT (Actix, Hyper) nebo Apache License 2.0 (Actix, Atreugo, ASP.NET Core, Firenio, Spring).

C.6 Frameworky pro vývoj webových aplikací

Využití frameworků může při vývoji webových aplikací přinést určité výhody, mezi které se řadí:

- lepší udržitelnost kódu, jeho rozdělení a znovupoužití,
- zrychlení vývoje webových aplikací.

Nevýhody využití frameworků jsou naopak následující:

- je nutné strávit určitý čas učením se syntaxe a způsobu užití vybraného frameworku,
- frameworky čelí riziku ztráty popularity, což znamená riziko ztráty podpory frameworku. [28]

Srovnání 3 velmi populárních frameworků Angular, React a Vue čerpá informace z prací [29] [30], u kterých bylo nutné pro zhodnocení obtížnosti učení se syntaxe a principů frameworků určité míry subjektivity.

Z hlediska zmíněné obtížnosti učení se vývoje pomocí jednotlivých frameworků je nejlépe hodnocen Vue následovaný frameworkem React a jako nejobtížněji naučitelný byl označen framework Angular. [29] [30]

V porovnání výkonu pomocí simulace používání webové aplikace implementované pomocí jednotlivých frameworků opět nejlépe obstál framework Vue. Následován byl frameworky Angular a React. [29]

Všechny 3 zmíněné frameworky jsou vydávány pod licencí MIT.

K vývoji konfigurační webové aplikace je také možno využít knihoven a frameworků, které zjednodušují tvoření vzhledu stránky a implementaci responzivního návrhu, jehož účelem je podpora velkého množství typů displejů a ovládacích prvků různých typů zařízení.

Příklady takových frameworků jsou Bootstrap, Foundation a Zurb, které jsou dostupné pod licencí MIT.

D Zdrojové kódy a další přílohy

Následující přílohy byly vloženy do IS EDISON.

D.1 Zdrojový kód nástroje pro stahování konfiguračních dat

D.2 Dokumentace zdrojového kódu nástroje pro stahování konfiguračních dat

D.3 Příklady zpracovaných konfiguračních dat

D.4 Zdrojový kód agregací použitých v MongoDB

D.5 Zdrojový kód API zpřístupňující konfigurační data

D.6 Zdrojový kód webové aplikace pro konfiguraci automobilů

D.7 Dokumentace zdrojového kódu webové aplikace pro konfiguraci automobilů